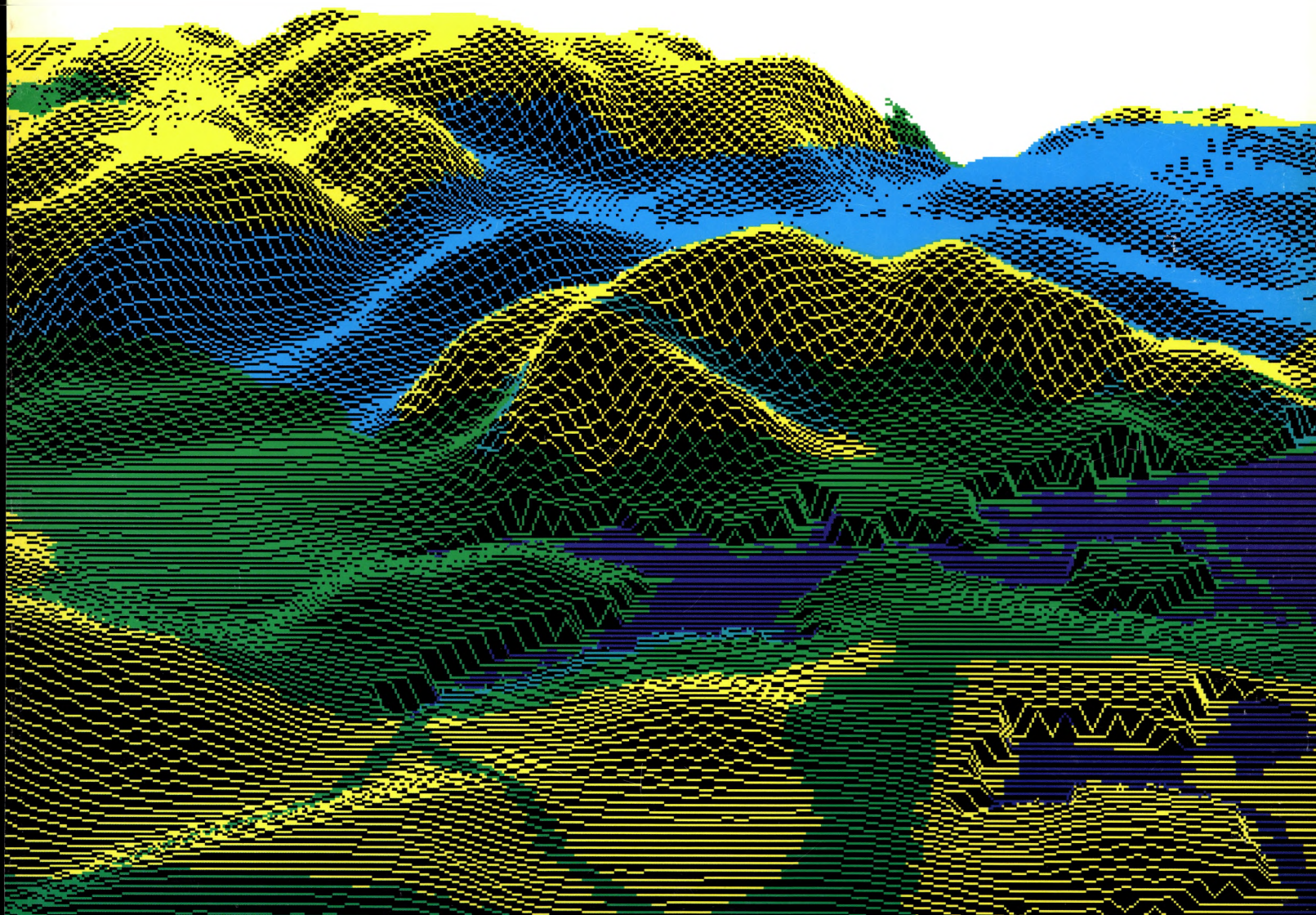
IBM Personal Systems Developer

A Publication of the IBM Developer Assistance Program
Fall 1990

- Multimedia & Graphics
- OS/2 1.3
- XGA



IBM Personal Systems Developer

Table of Contents

Editor's Comments 3

Developer Assistance Program

Developer Assistance Program Update 4

IBM Directions

Interview with Satish Gupta 7

Interview with Lee Reiswig 11

OS/2 Standard Edition

New Developments in 16-bit OS/2 13

Spotlight

AutoCAD Version 10 for OS/2 17

Multimedia & Graphics

IBM Multimedia Product Chronology 26

Programming the IBM M-Motion Video Adapter/A and M-Control Program 33

Spatial Analysis 40

CD-ROM and Multimedia 48

An OS/2 Business Graphics Toolkit 53

Corel Draw Migration to OS/2 PM 58

Collage PM: A Raster (Bitmap) Graphics OS/2 PM Application 68

XGA Benefits OS/2 Applications 79

XGA Technology 81

Programming the Audio Visual Connection 90

Software Tools

XIPC: A Comprehensive IPC Environment for OS/2 100

Front-ending Applications with EASEL 107

Extended Edition Database Manager

Configuring Remote Data Services 115

The Market

Market Update 124

© Copyright 1990 by International Business Machines Corporation. Printed in U.S.A.

IBM Personal Systems Developer is published by the Entry Systems Division of International Business Machines Corporation, Boca Raton, Florida, U.S.A., Dick Conklin, Editor.

Titles and abstracts, but no other portions, of information in this publication may be copied and distributed by computer-based and other information service systems. Permission to republish information from this publication in any other publication or computer-based information system must be obtained from the Editor.

IBM believes the statements contained herein are accurate as of the date of publication of this document. **However, IBM hereby disclaims all warranties either expressed or implied, including without limitation any implied warranty of merchantability or fitness for a particular purpose. In no event will IBM be liable to you for any damages, including any lost profits, lost savings or other incidental or consequential damage arising out of the use or inability to use any information provided through this publication even if IBM has been advised of the possibility of such damages, or for any claim by any other party.**

Some states do not allow the limitation or exclusion of liability for incidental or consequential damages so the above limitation or exclusion may not apply to you.

This publication may contain technical inaccuracies or typographical errors. Also, illustrations contained here may show prototype equipment. Your system configuration may differ slightly.

This publication may contain articles by non-IBM authors. These articles represent the views of their authors. IBM does not endorse any non-IBM products that may be mentioned. Questions should be directed to the authors.

This information is not intended to be an assertion of future action. IBM expressly reserves the right to change or withdraw current products that may or may not have the same characteristics or codes listed in this publication. Should IBM modify its products in a way that may affect the information contained in this publication, IBM assumes no obligation whatever to inform any user of the modification.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming or services in your country.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever.

All specifications are subject to change without notice.

To correspond with the *IBM Personal Systems Developer*, please write to the Editor at IBM Corporation, Internal Zip 4607, P.O. Box 1328, Boca Raton, FL 33429-1328.

Trademarks

IBM, Personal Computer AT, Personal System/2, PS/2, Operating System/2, OS/2, AIX, Bookmaster, PROFS, KnowledgeTool, Audio Visual Connection, and Micro Channel are registered trademarks of IBM Corporation.

PC XT, AS/400, Personal Computer XT Model 286, Systems Application Architecture, SAA, C/2, and Proprinter are trademarks of IBM Corporation.

COMDEX is a trademark of the Interface Group, Inc.

AutoCAD, Autodesk, AutoSketch, AutoSolid, AutoLisp, and AutoShade are registered trademarks and AutoFlix is a trademark of Autodesk, Inc.

Phar Lap is a registered trademark of Phar Lap Software, Inc.

National Geographic is a registered trademark of National Geographic Society.

SAS Institute is a registered trademark of SAS Institute.

DeScribe is a registered trademark of Lenname Advanced Products, Inc.

Ethernet is a trademark of Xerox Corporation.

UNIX is a registered trademark of American Telephone and Telegraph Corporation.

Microsoft, MS, MS-DOS, Xenix, and CodeView are registered trademarks of Microsoft Corporation.

Lotus and 1-2-3/G are registered trademarks of Lotus Development Corporation.

Macintosh and MacDraw are registered trademarks of Apple Computer, Inc.

Corel is a registered trademark of Corel Systems Corporation.

MOTIF is a registered trademark of Open Software Foundation.

Data General is a registered trademark of Data General Corporation.

PageMaker is a registered trademark of Aldus Corporation.

Intel, i486, i860, and ActionMedia are trademarks of Intel Corporation.

PostScript, Adobe Illustrator, Adobe Type Manager, and Adobe Systems Incorporated are registered trademarks of Adobe Systems Incorporated.

Cheltenham is a registered trademark of International Typeface Corporation.

XIPC is a trademark of Momentum Software Corporation.

VMS and Ultrix are registered trademarks of Digital Equipment Corporation.

TWA is a registered trademark of Trans World Airlines, Inc.

Delta is a registered trademark of Delta Air Lines, Inc.

Simware is a registered trademark of Simware, Inc.

Sony is a registered trademark of Sony Corporation.

Compugraphic is a registered trademark of Compugraphic Corporation.

Bitstream is a registered trademark of Bitstream, Inc.

Dataquest is a registered trademark of Dataquest Incorporated.

CASE:PM is a trademark of CASEWORKS, Inc.

Collage PM is a trademark and Inner Media is a registered trademark of Inner Media, Inc.

Hewlett-Packard and LaserJet are registered trademarks of Hewlett-Packard Company.

Sun Microsystems is a registered trademark of Sun Microsystems, Inc.

Multiscope and Logitech are trademarks of Logitech, Inc.

SPANS is a registered trademark of TYDAC Technologies, Inc.

EASEL is a registered trademark of Interactive Images, Inc.

Editor's Comments

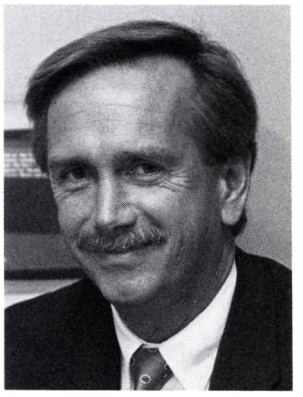
In this issue we're giving some overdue attention to an applications area that is well suited to OS/2: graphics and multimedia. IBM's new XGA support, which provides a big boost to software requiring higher resolution graphics and faster performance, makes the timing of this focus particularly appropriate. Our articles describe XGA hardware and various OS/2 graphics and multimedia applications from IBM and other companies.

We also look at our new and improved 16-bit OS/2 release, version 1.3, which fulfills our promise of a downsized OS/2 for 2 MB systems. Lee Reiswig, vice president of programming, offers some comments of his own.

We're pleased to announce the availability of subscriptions to the *IBM Personal Systems Developer*. Previously, most copies have been distributed through a controlled circulation; now, subscriptions can be entered as with any magazine—by telephone, mail, check, or credit card. Have your copies disappeared while making the office rounds? Now your co-workers can receive their own copies! You'll find some tear-out subscription cards in this issue.

Missing a back issue? During our first two years, we published dozens of useful articles that are now out of print. Not to worry. There's a single place where you can find all of them. A new book from Microsoft Press, *OS/2 Notebook: The Best of the IBM Personal Systems Developer*, provides a bound reference of articles, arranged by topic, from issues 1 through 7. Available in November 1990, this book (ISBN 1-55615-316-3) will sell in bookstores for $29.95.

Dick Conklin, Editor



Dick Conklin

Blue icons indicate a technical article.

Yellow icons indicate a marketing article.

Developer Assistance Program

Developer Assistance Program Update

Don Barovich

by Don Barovich

The IBM Developer Assistance Program provides a variety of support services for companies who are writing OS/2 applications for sale.

1. **Technical Support**: Support via IBMLink electronic interface to IBM's support organization. It is available twenty-four hours each day, seven days a week.

2. **Defect Support**: A toll-free number to report defects and receive the latest corrective service diskettes.

3. **Software Discounts**: 40% discount on selected OS/2 products ordered directly from IBM.

4. **Hardware Rebates**: 5–10% rebates on selected PS/2 products purchased from IBM authorized dealers.

5. ***IBM Personal Systems Developer*:** Free subscription to this quarterly technical publication.

6. **Code Downloading Service (ESDTOOLS):** Access to sample programs, demonstration packages, and other productivity aids. Some sample programs and listings contained in the *Developer* will be available through this service.

7. **Technical Education**: Invitations to participate in IBM-sponsored technical seminars and workshops.

If you are not a member of the IBM Developer Assistance Program, you can request an enrollment form (described below).

ELIGIBILITY AND ENROLLMENT

The IBM Developer Assistance Program is for software developers working on products for commercial release. You are eligible to participate in the program if:

- You develop products that support IBM OS/2 or IBM DOS.
- Your company is a U.S. company or the U.S. subsidiary of a non-U.S. company.
- You are currently marketing your products.

If you are not currently marketing a product, you can submit a non-confidential, detailed business plan showing marketing and development activities and schedules. IBM reserves the right to accept or reject an application based on this business plan. There is no charge for enrollment in the Developer Assistance Program.

To request an application form or ask questions about the program, contact Developer Support:

IBM Corporation
Developer Support
Internal Zip 4607
P.O. Box 1328
Boca Raton, FL 33429-1328
(407) 982-6408

OS/2 CONVERSION WORKSHOPS

The IBM Developer Assistance Program is now offering three *free* OS/2 Conversion Workshops:

Workshop 1	Converting a DOS Application to OS/2
Workshop 2	Developing a Presentation Manager (PM) Interface for an OS/2 VIO Application
Workshop 3	Converting a Windows 3.0 Application to OS/2 Presentation Manager

General Information

Workshop Format: The workshops combine lectures and lab time. Lectures, in a classroom setting, focus on the specific workshop topic. A laboratory next to the classroom contains workstations for attendee use during workshop lab sessions and evenings. When in the lab, attendees will work on their applications, using the information learned in the classroom lectures. The lab remains open each evening until 10:00 PM (technical assistance is available until 6:00 PM).

Attendance: By invitation only. For qualified companies, up to two attendees per company, working as a team, may participate. Seating is normally limited to twelve programmers per session.

Location: The workshop is located in the Service Center (South Tower), next to the lobby of the Royce Hotel, in West Palm Beach, Florida. Attendees should plan to stay at the Royce Hotel (the hotel provides free limo service to/from Palm Beach International Airport; auto rental is not recommended).

Financial Considerations: Each workshop, including Continental breakfast and lunch, is *free of charge*. Attendees are responsible for all other expenses (hotel, travel, dinner, incidentals, and so on).

For additional information about this extraordinary opportunity from IBM, please contact Jean Hicks (407) 982-0165 or Marilyn Johnson (407) 982-5514.

Workshop 1. Converting a DOS Application to OS/2

Description: A five-day, intensive, "hands-on" workshop designed to help you port your commercial DOS application program to OS/2 Standard Edition (SE) and get it running in protected mode in a PM text window.

Objective: To give you a "jump-start" on the task of moving your DOS application(s) to OS/2 and to help reduce the time needed to get your application(s) to market. We cannot guarantee that you will finish porting your application in one week. However, you should have a significant portion completed, and leave with the knowledge necessary to complete the job.

Lecture Topics: OS/2 Philosophy, Terms, and Concepts; Architectural Differences; Int 21; Processes; Threads; OS/2 File System; Interprocess Communications; OS/2 Application Programming Interfaces (APIs); Signals; Dynamic Link Libraries (DLLs); Device Monitors; Programming to the Device Driver API; PM Development; Protect-Mode Debugging.

Prerequisites: Attendees should be lead programmers with a strong DOS programming background and intimate knowledge of the application to be ported. Attendees should understand C language.

Preparation: Attendees should be able to perform typical "user" tasks with OS/2. We also recommend that the attendee study one of the commercially available texts on programming for OS/2 before attending this workshop. Previous experience with multitasking operating systems will also be helpful.

Prerequisite Application: Attendees must bring to the workshop source code for a commercial DOS application. The program must be coded in a language that is compatible with OS/2. Bring all supporting software your application will need, such as libraries, editors, compilers, debuggers, and so on. If your program has unique requirements (database, communications, Microsoft Windows, direct interface with hardware, coded in a language other than C), please discuss these considerations with your workshop contact before committing to attend the workshop.

The IBM Developer Assistance Program can be reached at (407) 982-6408.

Workshop 2. Developing a Presentation Manager Interface for an OS/2 VIO Application

Description: A five-day, intensive, "hands-on" workshop focused on developing programs for the OS/2 SE PM.

Objective: To assist you in developing a running PM interface for your OS/2 text window (full-screen) application.

Lecture Topics: PM Philosophy, Terms, and Concepts; Features; Menus and Accelerators; Creating/Managing Windows; Getting/Dispatching Messages; Graphics; Fonts; Working with Text; Using the Keyboard and Mouse; Using Thread Semaphores and the Timer; Help Manager; Introduction to the Dialog Manager; Building PM Programs.

Prerequisites: Attendees should be lead programmers, have a thorough knowledge of their application, and have attended a previous session of the base workshop (Workshop 1 above). Equivalent OS/2 knowledge may be sufficient in some cases. Attendees should be able to program in C language.

Preparation: Attendees should study a commercially available text on programming for OS/2 before attending this workshop.

Prerequisite Application: Attendees must bring to the workshop source code for a commercial application that operates in an OS/2 text-mode window. This will normally be the application converted to OS/2 during the prerequisite workshop. Also, bring all supporting software your application will need, such as libraries, editors, compilers, debuggers, and so on. If your program has unique requirements, please discuss them with your workshop contact before committing to attend this workshop.

Workshop 3. Converting a Windows 3.0 Application to OS/2

Description: A five-day, intensive, "hands-on" workshop designed to help you port your commercial DOS Windows 3.0 application program to OS/2 SE PM.

Objective: To give you a "jump-start" on the task of moving your Windows application(s) to OS/2 PM and to help reduce the time needed to get your application(s) to market. You may not finish porting your application in one week, but you should have a significant portion completed and leave with the knowledge necessary to complete the job.

Lecture Topics: Differences between OS/2 and Windows (messages, memory management, the 'main' function, scheduling, the help manager, resources, compilation and linking, and so on); the Software Migration Kit—what is does for you and how to use it; Multitasking (preemptive scheduling threads, processes); Dynamic Linking; Object Orientation; Performance; How to Design Portable Code; the Graphics Interface; the VIO Subsystem; and SAA (Common User Access and Common Programming Interface). In addition, you will learn about tools, such as CASE:PM, that can greatly reduce your conversion effort.

Prerequisites: Attendees should be lead programmers with a strong DOS and Windows programming background, and intimate knowledge of the application to be ported. Attendees should understand C language. OS/2 and PM knowledge will be helpful, but are not mandatory.

Preparation: Attendees should be able to perform typical "user" tasks with OS/2. We also recommend that attendees study one of the commercially available texts on programming for OS/2 before attending this workshop. Previous experience with multitasking operating systems will also be helpful.

Prerequisite Application: Attendees must bring to the workshop source code for a commercial DOS Windows 3.0 application (the program must be coded in C language). Bring all supporting software your application will need, such as libraries, editors, compilers, debuggers, and so on. If your program has unique requirements (database, communications, direct interface with hardware, coded in a language other than C), please discuss these considerations with your workshop contact before committing to attend.

Don Barovich, *IBM Entry Systems Division, 1000 NW 51st Street, Boca Raton, FL 33429.*
Mr. Barovich is manager of the Developer Assistance Program and Developer Support Strategy. He has been with Entry Systems Division since it was formed and has worked on PC software since 1982. He has a BS in Electrical Engineering from Penn State University.

Interview with Satish Gupta

Developer: What is your role at IBM?

Gupta: As product manager for Digital Video Interactive (DVI) products, I am responsible for the development and management of products that integrate digital video and audio into the PC. Besides the recently introduced ActionMedia 750 DVI application development platform, I am responsible for the development of the next generation of DVI hardware and software (OS/2) and also the joint development DVI project with Intel. My group is part of the Entry Systems Division, which also has the development responsibility for the Audio Visual Connection (AVC) and M-Motion Video Adapters.

Developer: What is IBM's strategy for multimedia?

Gupta: IBM intends to provide a full range of multimedia capabilities on the PS/2 and OS/2 platform. Our current offerings start with AVC and Storyboard with still image and audio capabilities, and go all the way to digital full-motion video, high-quality audio, and communications with servers. OS/2 is our primary platform, but we also will support other operating systems. Besides enhancing the AVC to support additional multimedia capability, we will work with independent software vendors to encourage development of authoring tools and applications. As the cost of audio and video technology allows, we will move these multimedia capabilities onto the motherboard of the PS/2. We expect that soon, when PS/2s have built-in multimedia capabilities, applications developers will employ them in their user interfaces. For example, when a message is most appropriately given in voice, the application designer will call on the voice capability of the PS/2. If applications can use three or four senses to get their messages across, they will be more effective at communicating with their users.

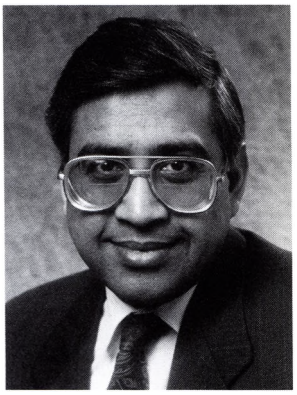
Developer: What multimedia products does IBM currently have on the market?

Gupta: AVC is available on both OS/2 and DOS. It is an authoring tool that allows you to develop applications around high-quality audio and photo-realistic images, integrated with other OS/2 applications. Integrated means that you can use it on a Local Area Network (LAN) or with other standard DP applications. (See "Programming the Audio Visual Connection" in this issue.)

The M-Control Program is a set of Presentation Manager (PM) controls to make it easy for developers to use laser disc-based audio and video. Applications also can use audio that is not coming from the laser disc. (See "Programming the M-Motion Adapter/A and M-Control Program" in this issue.)

Although we have both DOS and OS/2 tools, we find that many of our customers prefer using the OS/2 versions.

The ActionMedia 750 for DVI is another recent product. Our objective is to get the DVI technology to our customers as fast as possible. DVI is implemented in a microprogrammable circuit that will evolve toward higher performance and lower cost. Several tools are available to facilitate the use of DVI technology.



Satish Gupta

IBM intends to provide a full range of multimedia capabilities on the PS/2 and OS/2 platform.

Developer: How is OS/2 better than DOS for multimedia development?

Gupta: OS/2 has some key capabilities. Multitasking allows for audio, video, and communications to run simultaneously. Windowing is an important presentation technique. The 32-bit addressing speeds up multimedia applications. The communications capability of OS/2 allows these applications to interact with other PS/2s on a network or with a large multimedia database located on a mainframe. The OS/2 versions of AVC and the M-Control Program will continue to take full advantage of these OS/2 capabilities.

Developer: What is the relationship between multimedia and graphics?

Gupta: Many underlying technologies are the same, but the applications are different. They come from two extremes, and both will continue to exist on the same PC. Graphics is traditionally the result of the computer processing some type of model. For example, we can create charts from data contained in a spreadsheet. This has been advanced to a very sophisticated point. We now can create 3-D models with graphics and animation programs. Synthetic voice and Musical Instrument Digital Interface (MIDI) music are analogous to graphics in this sense.

Multimedia is changing the role of the PC from a computing tool to a communicating tool.

On the other extreme, there is information that people create in their daily lives, such as when talking, drawing pictures, and presenting or teaching classes. The most appropriate way to use that information on a PC is to capture it, not recreate it. The TV industry does this. The benefit is preserving the emotion, intonation, and richness of the image, and this is where graphics stops and multimedia begins. Multimedia applications are attempting to harness the PC to deliver that information in the most effective way.

Developer: What type of applications will benefit most from multimedia techniques?

Gupta: I believe the entire spectrum of personal computer applications will be affected. Multimedia is changing the role of the PC from a computing tool to a communicating tool. As this happens, the PC will become a more important element of the information delivery industry (such as publishing and broadcasting). The PC allows publishing to be more interactive. As another example, correspondence, represented today by word processing, could be greatly enhanced with voice and pictures. This evolution, however, will happen in stages. Today there are multimedia applications in training, accessing public information, and merchandising. The PC makes these applications more affordable, more effective, and more accessible. Next we will see traditional PC applications (word processing, spreadsheets, and so on) include multimedia as a natural extension of their user interfaces. Word processors and spreadsheets will have voice, still pictures, and motion pictures attached to text.

We will also see video databases. At PC Expo, IBM demonstrated the OS/2 super server technology delivering four full-motion video streams concurrently over a standard 16 MB Token Ring network to PS/2s.

Finally, graphics has become part of user interfaces today. When multimedia is part of the basic PC, developers will incorporate it into their user interfaces.

Developer: What other hardware will be needed for these multimedia applications?

Gupta: There will be a wide range of hardware, depending on the needs of the applications. To run an AVC application, a standard VGA display provides acceptable video. High-resolution displays are not necessary, but they will offer improved picture quality. If you need high-quality audio, add an audio adapter that will produce high-fidelity voice or music through hi-fi speakers.

If the application needs to capture pictures, then a camera or a scanner is needed. The camera can be attached to a capture adapter board that the AVC supports. VCRs will become an output recording device for multimedia applications that need portable output.

Developer: How will High Definition Television (HDTV) affect multimedia computing?

Gupta: The battle over standards for HDTV will prevent it from being an important component in multimedia applications in the next two to three years. When it does arrive, it will raise users' expectations for display resolution. In the PC these resolutions will be there before HDTV becomes important. One million pel PC displays are close to the reso-

lution of HDTV. The thrust of HDTV is to digitize the picture and audio. When that happens, we are likely to see TV sets that are also computers and computers that also function as TVs.

Developer: What kind of disk storage will be used by multimedia applications?

Gupta: CD-ROM was invented initially for audio and adapted to data storage. It is becoming popular because it is an inexpensive way to deliver large amounts of data (over 600 MB). CD-ROM will continue to play an important role in multimedia publishing applications (documentation, catalogs, tutorials, and reference documents). The PC helps the user to search these large collections of information and to make these documents interactive.

I see hard disks continuing to be used with multimedia spreadsheets, word processors, and databases. I expect the images and audio components of applications that deal with more personal data to fit within the capacities of magnetic disks. The hard disks will also be appropriate for multimedia databases that need frequent updates.

As erasable optical discs become more available, the role of optical storage will expand. Removable writable optical storage can be very useful for publishing applications.

Developer: Will video conferencing be a big application of multimedia PCs?

Gupta: I believe teleconferencing has not taken off because it is still inconvenient. Nobody wants to go to a special theater to conduct a meeting. Multimedia personal computing offers the potential for personal multimedia conferencing just like any other application, such as E-Mail. It will not be special or out of the ordinary; in fact, it should be as easy as the telephone. From your office, you could click on an addressee and be connected to another person. The exchange could involve pictures and audio over a traditional LAN circuit or high-speed telephone network. While motion video is important in some circumstances, much can be done with still images and audio. There is a large opportunity to improve on today's document-sharing technology (fax) by sending high-quality still images of documents. Technologies like DVI will allow us to ship a document very quickly between locations and enable that document to be easily viewed on a PC screen. The key to a broad acceptance of PC conferencing will be the usability and picture quality.

Developer: Could you give us an example of an interesting multimedia application?

Gupta: Sure. We are working with Worldspan, a consortium of TWA, Delta, and Northwest Airlines. They want to market products such as hotels, resorts, vacations, and cruises through travel agents. They want to provide compelling audio and video clips, interactively selectable by the travel agents as they work with the customers. Worldspan eventually plans to update the available product pool as soon as a new product is ready, and to interact with the airline reservation system to complete the sale. Finally, they may want to produce a video tape output for customers to take home after the sale. Worldspan has chosen OS/2 and PS/2 with DVI as the platform to implement this application.

Developer: What is IBM's position concerning the emerging multimedia standards?

Gupta: We know that, in the PC business, open standards are good for both suppliers and customers. IBM will support several emerging standards. There is a new standard proposed for CD-ROM by Sony and Microsoft called CD-ROM/XA. There is an International Standards Organization (ISO) standard for still picture compression/decompression called the Joint Picture Encoding Group (JPEG). Another standard being created by ISO called Motion Picture Encoding Group (MPEG) will mature in the next year or two. CCITT is working on a standard called PX64. We will support all these standards.

Because the DVI circuit is programmable, we can cost-effectively implement these standards in our products as they become fully defined. The programmability of our products allows us to be on the leading edge of standards implementation as they evolve over time.

IBM will support several emerging multimedia standards.

Developer: What advice would you give to developers wanting to get started with multimedia?

Gupta: If you want to be in the forefront, you must start now. My advice is to start by taking a couple of courses. IBM offers several courses for DVI application development, AVC, and briefings on IBM tools. Several companies offer support in the area of content production. Intel also offers applications development training for DVI. IBM courses are offered by our Multimedia Solutions marketing team in Atlanta, which supports both end users and developers. They can be reached at (404) 238-4646 (IBM Multimedia Solutions, 4111 Northside Parkway, Atlanta, GA 30327).

Acquire AVC and experiment with it. Learn from experience. Programming is not the challenge; the challenge is applying this new data type to real conceptual problems. Applying multimedia is new to most PC application developers. Fortunately, others—such as those in the television industry—have mastered its application. Their industry provides an existing skill base where we can go for instruction and support.

Developer: If an OS/2 application developer has some ideas for a product, and would like to become a business partner with IBM, where can they start?

Gupta: We are working on multimedia applications with over seventy developers today. The Multimedia Solutions group in Atlanta is the place to start. *[Editor's Note: See this issue's Market Update article for more information about IBM multimedia support programs.]*

IBM Directions

Interview with Lee Reiswig

Lee Reiswig is vice president of programming for IBM's Entry Systems Division. He is responsible for personal systems software strategy and development.

Developer: The new agreement between IBM and Microsoft has received a lot of attention; can you summarize it for us?

Reiswig: I'm very pleased with the way this turned out. IBM becomes the primary developer for both 16-bit and 32-bit versions of OS/2. Microsoft is the primary developer for DOS, Windows, and an OS/2 version that will be portable to other computer instruction sets, such as RISC (Reduced Instruction Set Computing). Each company will make development contributions to the other company's products, and cross-licensing is provided.

Developer: Who benefits most from this agreement?

Reiswig: I look at this as a real "win-win" situation. For our customers these improvements translate into better quality products and shorter development cycles. For both companies it brings the improved productivity and efficiency that comes from single-company, single-location development activities. For software developers it means getting technical support from a single source.

Developer: You claim the new OS/2 version 1.3 runs faster and uses less memory—two characteristics that usually conflict with each other. How did you pull it off?

Reiswig: It's a real tribute to the folks at our Boca Raton (Florida) Programming Center. They not only helped us meet our objective of a 2 MB version of OS/2, but in the process they found ways to significantly improve performance, too. This release runs up to 5% faster in a system with unconstrained memory, but up to 25% faster in a constrained environment. On a LAN, program loading is up to five times as fast. Much of the tuning was done by repackaging program segments so code that was infrequently used is no longer loaded when not needed. This was all accomplished with no loss of function.

Developer: But what does that mean to the entry-level OS/2 user?

Reiswig: It means that someone with a minimum 2 MB system can run a multiple-application OS/2 environment, typically something like two OS/2 programs and one DOS program, with good performance. This is particularly attractive for upgrading existing equipment to OS/2 since the cost to upgrade is low—in fact, most of our system units are sold with 2 MB of memory on the motherboard.

Developer: How did the changes impact disk storage requirements?

Reiswig: We reduced disk requirements to the point where we can run OS/2 Standard Edition (SE) 1.3 comfortably on a system with a 30 MB fixed disk. This was done by shrinking the size of the files themselves, and making the installation process more selective so that only the needed files are disk-resident. Total disk usage now runs between 7 and 11 MB—down from 13 MB. A typical user will require 9 MB.

Developer: What if users later change their minds and want to add part of OS/2 that was not installed originally? Doesn't that mean reinstalling everything?

Reiswig: No. The reinstallation process for OS/2 SE 1.3 is also selective, so you only add what you need.

OS/2 1.3 runs up to 5% faster with unconstrained memory, and up to 25% faster in a constrained environment.

We have a whole new print spooler, several new and improved printer drivers, and the driver installation procedure has been streamlined.

Developer: OS/2 has received criticism because of the lack of printer drivers, difficulty in installing them, and problems with the print spooler. Has this support been improved?

Reiswig: Yes. We have a whole new print spooler, several new and improved printer drivers, and the driver installation procedure has been streamlined. Printer installation and setup has been integrated in one place to provide a more intuitive solution.

Developer: Is this the first release to include the Adobe type fonts you announced earlier in the year?

Reiswig: Adobe type fonts appear in OS/2 1.3 for the first time; you'll notice an improvement in text quality, choice of sizes, and font selection. Fonts on the display and printed page are identical, so now applications can really be what-you-see-is-what-you-get. Additional fonts can be added later.

Developer: Last fall IBM and Microsoft positioned DOS, DOS+Windows, and OS/2. Since then, Windows 3.0 and OS/2 1.3 have appeared on the scene. Has this changed your strategy?

Reiswig: Not at all. When we made the COMDEX (Fall '89) statement, we were aware of the planned Windows 3.0 delivery. OS/2 1.3 fulfills the commitment we made then to work toward a 2 MB entry into OS/2. It offers all the previous functions and advantages of OS/2, such as preemptive multitasking and OS/2 Extended Edition (EE) extensions, plus the new features I just mentioned. This makes it the right choice for a high-function graphical workstation. Since there is some overlap between Windows and OS/2, you can't look at the physical requirements alone. You have to look at the kinds of software applications you plan to run on the workstation, and beyond that, its role in the business—connectivity, shared data, and so on.

Developer: You also said last fall that the Database Manager and Communications Manager components of OS/2 EE will someday be offered as a separate product. Can you elaborate more on that?

Reiswig: We are still moving in that direction, although we don't have a specific product offering yet. We are planning to have the OS/2 EE-exclusive function available for both IBM and OEM versions of OS/2.

Developer: OS/2 has been criticized as an operating system with few applications. When do you think it will have an acceptable number of applications in the eyes of most users?

Reiswig: We know of over a thousand commercial OS/2 applications today, and more are added every week. Some of the better known are Lotus 1-2-3/G, AutoCAD, and PageMaker. These are key Presentation Manager applications that help justify the move to OS/2.

In many customer accounts, OS/2 has been doing very well, especially the Extended Edition. When distributed systems and connectivity are important, it is extremely competitive. Our support for mixed environments now encompasses software protocols like Ethernet and various OEM hardware platforms. Large corporations are developing their own OS/2 applications for in-house use, but of course you don't see those listed in any catalogs.

Developer: Where does this leave DOS?

Reiswig: DOS is still alive and well, and will continue to be a viable platform for many users.

OS/2 Standard Edition

New Developments in 16-bit OS/2

by David Reich

IBM has been working on 16-bit OS/2 to make it a "leaner and meaner" system. In the past, OS/2 has been criticized for requiring too much memory and disk space and not being fast enough. OS/2 1.3 addresses these issues by requiring less disk and memory, providing application and operating system speed increases, and adding new function for all machine sizes, without removing any function.

IBM's objective for OS/2 is for it to run acceptably in a system with 2 MB of RAM and a 30 MB disk. OS/2 1.3 will realize those goals and more. With new function added and none removed, OS/2 runs with good performance in this minimally configured system.

Every facet of OS/2 was examined during the development of OS/2 1.3 to identify any areas that could be improved—from installation to memory management. Figure 1 shows the more pronounced enhancements.

OVERALL ENHANCEMENTS

Memory Requirement: Significant memory reductions are expected, but specifications were not available at the time this article was written.

Performance: The operating system itself has been restructured. Additional parts of the system have been performance tuned to load faster and manage themselves more efficiently. In addition, the program loader now loads applications faster, especially over a Local Area Network (LAN). Management of the swap file and the swapper program itself have new enhancements: swapping is now smarter and faster.

Disk Space: The operating system now consumes significantly less disk space. This is because of the way it generates executables and the Selective Install, which allows a user to install only the required parts of the operating system. Additionally, other files that are not required to run the system in day-to-day operations are no longer installed on the hard disk.

The tuning and swapping enhancements save disk space. The swapper file is smaller because of more efficient swapping.

PM Windows Limit: Previous releases of OS/2 allowed only 1,200 windows in the entire system, which sometimes created problems for users of many concurrent, large applications. With OS/2 1.3, the limit is now over 15,000.

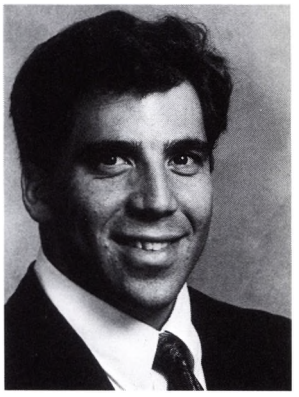
Performance Enhancements

- Enhanced Program Loader
- Swapper Enhancements
- Screen Switch Repaint Performance
- Kernel and Subsystem Tuning
- Selective Install

New Functions

- Enhanced Print Spooler
- Enhanced Printer Driver Installation
- Added Font Support
- New Presentation Manager Controls
- Drag-and-Drop Message Protocol

Figure 1. Summary of OS/2 1.3 Enhancements



David Reich

Every facet of OS/2 was examined during the development of OS/2 1.3.

PROGRAM LOADER

The loader code has been enhanced significantly in several areas. The first, and perhaps most noticeable, change is the smaller size of the system's executable files (.COM, .EXE, and .DLL). Specific linker options were used to compress the size of these files; code and data changes were made to make them more efficient. This directly affects system and application startup speed. Since smaller segments and files are being read, there is a significant reduction in disk I/O.

The other major enhancement to the loader is in how applications are read from disk. In the past, the loader executed many small file reads to load segments into memory in a specific order. In performance benchmarks, the OS/2 developers discovered another method of reading in segments from the disk that presents a significant savings in both disk accesses and load time. These new algorithms currently are being evaluated for patent filings.

A major impact of the loader enhancements is not only how much faster programs are loaded, but how much faster they are loaded over a LAN. All file accesses over a LAN are expensive. These new algorithms reduce the number of accesses required to load a file and thus make LAN program loads less expensive.

> *"The overall performance of OS/2 1.3 is excellent. In addition, this compacted version will enable software developers to offer more applications that take advantage of OS/2's broad range of functionality and reliability."*
>
> —Gordon E. Eubanks, Jr., president and CEO, Symantec Corporation

SWAPPER ENHANCEMENTS

OS/2 1.3 uses swapping to provide more virtual memory than is physically available in the machine. To make room for a requested piece of code or data, OS/2 1.3 writes segments of memory not currently being accessed to a special file on disk (called *swapping* to the swap file). When OS/2 is run in an overcommitted memory situation, performance naturally decreases because not all parts of an application reside in memory.

> *"Dramatically improved networking performance and reduced memory requirements will significantly improve the attractiveness of OS/2 for high-end business applications like the SAS System."*
>
> —Dr. Jim Goodnight, president, SAS Institute Inc.

Benchmark studies by IBM in Boca Raton found that the swapper always started with a small swap file and immediately began to expand it. They also found that the swapper transferred many small segments to and from the disk, taking almost as long as large transfers.

When OS/2 is started, the swapper (SWAPPER.EXE) reallocates any existing swap file (SWAPPER.DAT) and creates a new one. While doing this in OS/2 1.3, the swapper looks to see how much memory is in the system, and allocates a swap file of a reasonable size for the particular memory configuration. It is less expensive to make a larger swap file initially, when new items are expected to be added, than to reallocate it continually.

Another major improvement in the swapper is how items are chosen for swapping. When swapping a small segment, the swapper caches it in a staging area instead of writing it to disk. After caching several small segments, if the swapper finds that the staging area is full, it writes all the cached segments to disk with one write operation. This reduces the number of swapper-initiated disk writes. Since some cached segments will be reclaimed before they are written to disk, the number of swapper disk reads is further reduced. By reducing the number of disk operations, swap caching realizes significant gains.

SCREEN SWITCH REPAINT PERFORMANCE

Significant improvements have been made to repaint performance when switching between full-screen DOS and Presentation Manager (PM) sessions. Previously, when switching on a 2 MB system, PM asked each window to repaint itself. Under this memory-constrained system, this repaint could take a long time and could be perceived as a usability distortion to the user.

The OS/2 Video Subsystem now assumes the task of saving and restoring the entire graphical image of the PM screen group. Each memory segment used to hold the image is marked Least Recently Used (LRU) when the image is no longer required. This offsets the extra memory used for holding the image. This also makes these segments likely candidates to be swapped out, which dramatically improves performance.

> *"I especially like the improvements in print drivers. I like the way the LaserJet driver works."*
>
> —John McFetridge, director of technical marketing, Corel Systems Corporation

KERNEL AND SUBSYSTEM TUNING

The operating system as a whole has further improvements. In the past, many parts of the subsystems (Mouse, VIO, and so on) were already swappable. By restructuring some of the code, more parts of the system are now marked swappable, thus saving even more memory.

Besides being marked swappable, many of these segments are discardable. First, this saves memory, especially in a lower memory situation, where more swapping occurs. Another less obvious gain is in disk space usage. More code is now discardable. When a segment fault occurs and a victim is selected to be removed from memory, it is desirable to have a discardable segment. It does not have to be written to the swap file. The area of memory can simply be marked as unused, so the new data can be written into it. This helps to keep down the swap file size and to increase performance, since it requires fewer disk accesses.

SELECTIVE INSTALL

Selective install describes the new, smarter installation program for IBM OS/2. In previous releases, the user could choose several items while installing OS/2. Now, the user can make much finer choices. That is, the user can install the default system (as in previous releases) or choose each option to be installed. For example, OS/2 1.3 allows the user to choose individual fonts instead of getting all of them by default. A user can return at any time and install an option that was declined when the initial install was performed. It is not necessary to reinstall the entire system to add features.

Using selective install, the user can significantly reduce the disk space taken by OS/2. This can be particularly valuable on an entry-level 30 MB hard disk system. As new features are added to OS/2, this selective install will offer users increasing flexibility in creating systems optimized for their hardware. Of course, the user can always select the defaults and get the entire system.

> *"The presence of the Adobe Type Manager (ATM) means our users finally have true WYSIWYG. Users type as fast as they want and ATM generates screen fonts as fast as they can type. What they see on the screen appears on paper when they print."*
>
> —Allan Katzen, president and CEO, DeScribe, Inc.

NEW FUNCTIONS

In addition to the performance gains, OS/2 1.3 brings the following new functions:

Printer Installation: To use a non-default printer (such as the Hewlett-Packard LaserJet or IBM 4019 printer), the user must install a printer device driver. In previous versions of OS/2, installing a printer device driver meant using a combination of the OS/2 Control Panel and the Print Manager to install the driver, create a print queue, and hook up the two. OS/2 1.3 comes with a printer installation option in the Print Manager window. The user simply selects the desired printer driver. The installation tool does all the work of installing the driver.

OS/2 1.3 comes with a printer installation option in the Print Manager window.

Font Support: The new release of OS/2 will also have enhanced font support. This function improves the drawing speed of vector fonts. It enhances printing so that these fonts, as printed, will more closely match what the user sees on the screen called WYSIWYG—What-You-See-Is-What-You-Get.

Printer Drivers: Great strides have been made in increasing their dependability and functionality, but specifics were unavailable at the time this article was written.

SUMMARY

Much recent speculation about OS/2 is incorrect. The 16-bit OS/2 has not stagnated, nor has IBM stopped development work on it. On the contrary, OS/2 is coming of age very quickly. In less than three years, it has become a mature, lean system that can hold its own against competitive platforms.

> *"The development of OS/2 1.3 hastens the inevitability of OS/2 becoming the dominant operating system for corporate computing."*
>
> —Jim Manzi, president, Lotus Development Corporation

The author wishes to thank members of the IBM 16-Bit OS/2 Development Team who have contributed to this article: Paul Brightly, Bob Chernow, Edd Doutre, David Lanning, Tatchi Lay, Eric Rasmussen, and Brian Ross.

David E. Reich, *IBM Entry Systems Division, 1000 NW 51st Street, Boca Raton, FL 33429. Mr. Reich is a senior associate programmer working in the Boca Programming Center Build and Integration area, and has been working for IBM for three years. He has co-authored a book on OS/2 entitled "OS/2 Presentation Manager Programming," published by John Wiley and Sons.*

Would you like to get your own copy of the IBM Personal Systems Developer? Subscribe Today!

Yes!

I wish to subscribe to the
IBM Personal Systems Developer
for 1 year (4 issues at $39.95 per year).

NAME (please print) TITLE

COMPANY

ADDRESS

CITY STATE ZIP

☐ Bill me

☐ Check enclosed *(make check payable to Graphics Plus/IBM Developer)*

Charge my credit card: ☐ Visa ☐ MasterCard ☐ American Express

Card No. _____ Exp. Date _____

Signature _____

TELEPHONE ORDERS
1-800-READ-OS2

Yes!

I wish to subscribe to the
IBM Personal Systems Developer
for 1 year (4 issues at $39.95 per year).

NAME (please print) TITLE

COMPANY

ADDRESS

CITY STATE ZIP

☐ Bill me

☐ Check enclosed *(make check payable to Graphics Plus/IBM Developer)*

Charge my credit card: ☐ Visa ☐ MasterCard ☐ American Express

Card No. _____ Exp. Date _____

Signature _____

TELEPHONE ORDERS
1-800-READ-OS2

Yes!

I wish to subscribe to the
IBM Personal Systems Developer
for 1 year (4 issues at $39.95 per year).

NAME (please print) TITLE

COMPANY

ADDRESS

CITY STATE ZIP

☐ Bill me

☐ Check enclosed *(make check payable to Graphics Plus/IBM Developer)*

Charge my credit card: ☐ Visa ☐ MasterCard ☐ American Express

Card No. _____ Exp. Date _____

Signature _____

TELEPHONE ORDERS
1-800-READ-OS2

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1956 BRIDGEPORT, CT.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Personal Systems Developer
c/o Graphics Plus, Inc.
640 Knowlton Street
Bridgeport, CT 06608-9985

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1956 BRIDGEPORT, CT.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Personal Systems Developer
c/o Graphics Plus, Inc.
640 Knowlton Street
Bridgeport, CT 06608-9985

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1956 BRIDGEPORT, CT.

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Personal Systems Developer
c/o Graphics Plus, Inc.
640 Knowlton Street
Bridgeport, CT 06608-9985

Would you like to get your own copy of the IBM Personal Systems Developer?

Subscribe Today!

Spotlight

AutoCAD Version 10 for OS/2

Autodesk® develops, markets, and supports a family of computer-aided design, engineering, and animation software products for personal computers and workstations. According to Dr. Malcolm Davies, vice president of marketing and sales, "Autodesk established a worldwide standard for CAD software, as represented by over 315,000 current AutoCAD customers and a 63% market share on the DOS platform."

AutoCAD for OS/2 has been well received by users and the press. *PC Magazine* and some CAD specialty magazines have given it very good reviews. Some have said that AutoCAD by itself is enough to justify making the jump to OS/2.

Autodesk has a multi-platform product strategy, although today 80% of its revenue comes from the extended DOS version of AutoCAD. All versions of AutoCAD are built from a common core of code that is platform independent. The company has maintained this core code for many years. Ninety five percent of the OS/2 product is core code. This approach contrasts with Lotus' development of 1-2-3/G from scratch (described in the Summer issue of the *Developer*).

Product Summary

Name:	AutoCAD®
Category:	CAD
Price:	$3,000 $250 exchange from another platform
First Ship:	January, 1990
Suggested: Memory:	6 MB recommended
Other Platforms:	DOS, Extended DOS (Phar Lap®), UNIX® (Sun OS), VMS, Aegis (Apollo), Xenix®, Macintosh®, ULTRIX® (DEC).
International:	Sold in over 75 countries

OS/2 DEVELOPMENT

AutoCAD has roots back to a DOS program written before 1982. The OS/2 porting project started in November 1987, just after OS/2 1.0 shipped. Since AutoCAD is a very large program, OS/2's large memory support attracted the attention of Autodesk developers. Much of the internal design of AutoCAD for OS/2 was implemented during the availability of OS/2 1.0. When the Presentation Manager (PM) became available, a Graphical User Interface (GUI) was added. AutoCAD for OS/2 1.1 shipped in January 1990. The engineering team was so excited and proud of AutoCAD for OS/2 that they went on a marketing road show to talk to dealers and user groups to explain its benefits.

OS/2's large memory support attracted the attention of Autodesk developers.

At the same time, the company shipped a new Dynamic Link Library (DLL) that supports OS/2 1.2 and yields about a 10% performance improvement. It does not yet take advantage of all the OS/2 1.2 features. For example, it provides only limited support for long file names, and it does not support extended attributes.

THE PORTING PROCESS

The core code was not difficult to port because in 1985, Kern Sibbald, chief engineer, ported it to run under UNIX on Sun workstations. This UNIX code formed the basis of the OS/2 version. The developers added a

few items from the then-current DOS version and the Macintosh version. The switch from DOS to OS/2 was easy because many DOS system calls have direct OS/2 counterparts.

Major Components

An overview of AutoCAD for OS/2 is shown in Figure 1. Ninety-five percent of the platform-independent "core" code is common to all platforms. There are some #IF DEFs in the core, associated with variations such as byte order, plus a few modifications for DOS to look at memory. Until recently, Autodesk added to AutoCAD only features that could be supported by all platforms. With graphical user interfaces, Autodesk is starting to add features that are specific to the platforms. They still make their best efforts to keep all the features similar across platforms.

Ninety-five percent of the platform-independent "core" code is common to all platforms.

ACADLIB contains the platform-specific runtime services for AutoCAD, and is linked in with AutoCAD. On UNIX and OS/2, AutoCAD uses some libraries from the system, but for DOS, everything is there. For example, it includes the whole file library (fopen, fclose, and so on). The OS/2 team wrote this themselves so that, for example, fopen could have many more options (paged or temporary, and so on). ACADLIB contains many compiler dependencies and #IF DEFs to handle machine differences, such as the varying number of arguments in printing functions.

The remainder of the code is called *machine-dependent code* ("machines" include UNIX, DOS, BSD/System 5, OS/2, and Macintosh). It contains the operating system interface to shell programs and performs serial port I/O, timing routines, time-of-day, error recovery, signals, and so on.

Another major part of the OS/2 effort was to separate device code as DLLs, including digitizers, plotters, and displays. These are platform independent because they use the services of the machine code to support serial port I/O. Once they were DLLs, they could not access data and subroutines in core code, so a call-back mechanism was added to allow them to get back to core code services. Other AutoCAD platforms now are using the interface defined for OS/2.

The AutoCAD OS/2 Engineering Team

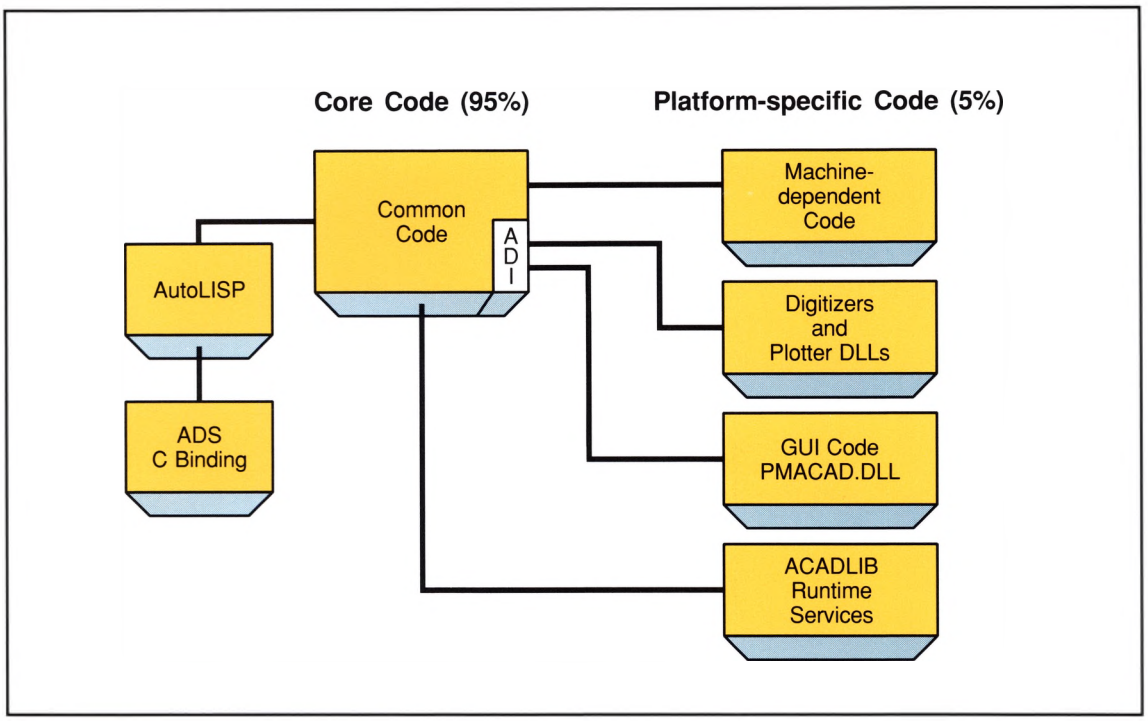


Figure 1. Block Diagram of the OS/2 Version of AutoCAD

PMACAD implements the GUI interface by putting up windows, managing them, and getting data.

AutoCAD has LISP language support built-in. LISP is an ideal language for processing lists of information similar to those that exist in an AutoCAD database. It runs as a separate process on UNIX and OS/2 platforms, and gives users and developers access to all AutoCAD commands. AutoLISP programs can access the AutoCAD database and select entities (such as lines, circles, and poly lines). Description of polylines can be very long lists. This language support is used by over 700 third parties and users to provide post-processing applications that use the AutoCAD database.

The Application Development System (ADS) is a C binding that allows developers to write C programs that can do anything that AutoLISP can do.

In UNIX versions, all the drivers (digitizers, displays, and plotters) are linked in, so the program can be very large. In DOS, the drivers are overlays. In OS/2, the developers decided to make the drivers DLLs. That was not easy, because all the drivers built into the core had to be moved out and then made available to the other outside modules. The capability of external drivers to interact with AutoCAD just like built-in drivers, an architectural improvement, is called the Autodesk Device Interface (ADI). ADI allows third parties to write drivers that can be used by AutoCAD. ADI is implemented differently in each platform. Under UNIX, the drivers talk across pipes; under OS/2 the drivers are DLLs.

AutoCAD contains three major threads.

DESIGN

AutoCAD contains three major threads, shown in Figure 2. As Sibbald explains, "Designing the threads in AutoCAD was interesting. Since AutoCAD is based on a DOS program, it is structured to go about its work and ask for user input only when it is ready. PM works in the opposite way. PM wants to be in charge of feeding input events to application programs. What we did was to leave the AutoCAD core with the I'm-in-charge view of the world. Rewriting the core would have taken an extraordinary amount of time; besides, our strategy is to maintain a large body of core code. Many people think that they must rewrite DOS applications to run under OS/2. They have taken that position because of developers' experiences with the Macintosh. AutoCAD for OS/2 proves that is not necessary."

All the windowing code was moved from the core into a thread called PMACAD.DLL, running at the input priority. PMACAD is a big PM message loop, handling keyboard input and output, the mouse, and graphics output. It never waits, so mouse movements and window activities are extremely responsive. Sibbald noted in comparison, "On UNIX, the mouse is sometimes jerky and the user cannot always grab a window and resize it instantaneously." AutoCAD gets all its input from a common memory buffer that is shared between threads one and two. The core code still thinks it is in charge, with a main menu and the user typing in text commands. The only modification made to core code to support this was to turn off the menus and to add subroutine calls to synchronize when the core text screen is allowed to be visible. "Even though the old AutoCAD program is still running, it looks beautiful to the user," said Sibbald.

Any developers starting today should write their core with multithreading in mind.

A third thread simply looks for activity from ACADL, the LISP application running as a child process. Since OS/2 does not provide signals when a child process dies, the developers created a thread to watch over it. If LISP dies, the thread will set a flag so that ACAD knows about it.

We asked Sibbald if future versions of AutoCAD would contain a core broken into more threads. "The core is based on DOS. That would take a re-architecture of AutoCAD, which we are working on. But that will take some time, and it is being done for a multitude of reasons," said Sibbald. "Economically, it doesn't make sense to rewrite a whole application for a platform that may be less than 5% of your business." He went on to say that any developers starting today should write their core with multithreading in mind.

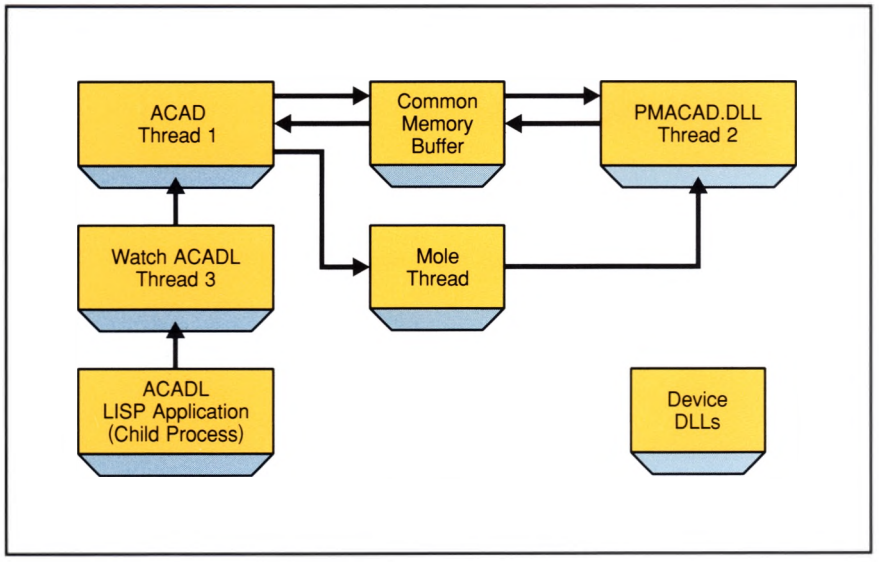


Figure 2. The Multithreaded Design of AutoCAD

CUA Guidelines

AutoCAD does not yet conform completely to IBM's Common User Access (CUA) guidelines. AutoCAD version 10 dialog boxes are in a PM frame, so the user can move them, but they are drawn by AutoCAD. This was done so that the OS/2 version could use the large body of core code for user interaction. AutoCAD OS/2 version 11 (available in 1991) will use entirely native PM dialog boxes. A constraint-based system that will be added to the core will work across the Sun, Macintosh, OS/2, and other platforms.

"Implementing these dialog boxes was relatively simple," Sibbald explained. "We put requirements for all our dialog boxes in an ASCII file. AutoCAD for OS/2 reads that information and formats the dialog boxes 'on the fly.' This system was done most easily on OS/2, so that's where we first developed it. Eventually we will change the core to support it, and then port the technique to other platforms."

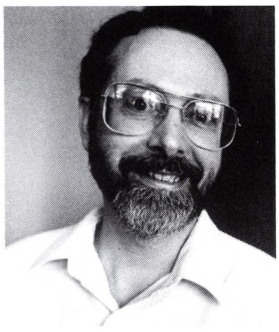
ORGANIZATION AND MANAGEMENT

The technical development team at Autodesk works on the Sun and Phar Lap versions of AutoCAD. They add all the features for the next release. Then two satellite porting teams move these features to their respective platforms. One team is responsible for UNIX and OS/2; the other does Macintosh development. These groups initially write the machine code for a new platform. After implementation, they primarily make changes in the GUI area and do not affect the core.

"OS/2 offers wonderful system services that we have yet to take advantage of. Because of our core code approach, if we want to use a system routine that is not available on other platforms, we have to write an equivalent routine and place it in core code for the other platforms to use," said Sibbald. "As a platform-specific development team, we can only change the core code to a limited extent

because the core team is continually enhancing it," added Steve Mighetto, a programmer who joined the team when the OS/2 1.1 effort began.

The Team Leader's Approach



Kern Sibbald

As we talked to the AutoCAD developers, it was apparent that the team functioned well together. We asked Kern Sibbald to describe his management style. "In any significant development, you meet many brick walls. I have an unusual way of dealing with them. My advice to the team was to identify the brick walls and then ignore them. It's amazing how often problems go away if you continue to work on other areas. When the brick walls don't vanish over time, we all get together and bulldoze them. The advantage of this approach is that we did not get stopped for weeks by a problem. I believe that when you concentrate on things you know how to do, things you don't know how to do start falling into place.

Sibbald admitted that he has a slightly unusual way of running projects. "Our team meetings have a very rigid format. The first two meetings I run with an iron fist; after that they run by themselves. We meet once a week and each developer gets five minutes to talk. No questions are allowed, unless for clarification. That way if anyone wants to question something, they must do it during their own five minutes. I have a list of tasks that must be done, but I generally let the team members choose what they want to do. If I have to make assignments, I try to choose things on which I think the programmers would like to work."

Sibbald believes in breaking tasks into little pieces. "This is an effective way to overcome brick walls. AutoCAD is huge—1.5 MB of code—and it can become unmanageable unless it is attacked in little pieces."

"We had aggressive goals. I was the one who chose the end date, and nobody said it couldn't be done. At every checkpoint, I reviewed where we were. The schedule was tight, but realistic," said Sibbald.

The Team's Point of View

At this point in our interview, Gary Scott, who specializes in internals and peripheral interfaces, said "Kern talks about the controlled meetings. However, this project was successful because it was a lot of fun. We all work well together."

Sibbald responded, "As the project leader, I was very excited about the project. That's contagious. In addition, each of us has a machine at home. When you work in an office, there are times when you say, gosh, I'd really like to try this. During the project, we spent many hours banging away at home on weekends. I work at home most of the time, coming into the office only occasionally. The other thing that made the project fun was that this was an important port that produced a very nice product. It was the best port the company has done."

"You can't put programming down at the end of the day and not pick it up until you are back in the office. At least I can't," said Scott. "Sometimes I watch TV while a compile or link is taking place."

Phil Ford, who joined Autodesk in 1989 to work on Dynamic Data Exchange (DDE) and the clipboard, also joined the discussion. "The project was fun because the scheduling was realistic and OS/2 is fun to use. Threads are brain teasers. This project was very satisfying because we did things that cannot be done on DOS," he said.

Steve Mighetto was impressed by how much visual feedback you can get from just a few lines of code. "For all of us but Kern, this was our first GUI project. By writing twenty lines of code, you can completely change how the application looks. That's fun," he said.

SPEED AND GUIs

GUI platforms offer certain advantages: the ease of use that comes from a common user interface style, the ability to have more than one application on the screen at a time, and added functionality. But there is a price in terms of speed.

"As the project leader, I was very excited about the project. That's contagious."

Other programs can interface with AutoCAD through the DDE code provided with the product.

The AutoCAD port to OS/2 points out one problem that is currently common to all graphical platforms: they are not as fast as programs that write directly to the screen. According to Sibbald, any windowing operating system results in about a 15%–20% performance hit. Most developers, however, feel that this problem will become less apparent to users as processor speeds increase.

Sibbald believes most AutoCAD users want speed, speed, speed. The majority are real power users. Many sit in front of AutoCAD all day and want it to be really fast. AutoCAD runs on OS/2 at nearly the same speed as the 640 KB DOS version. The Phar Lap version is 60% faster. This led to a discussion of speed improvement techniques that the team used. Phar Lap allows DOS applications to run in 32-bit flat address space in protect mode on 80386 CPUs.

PERFORMANCE TIPS

The GUI speed penalty is because the DOS version writes directly to the display hardware. Sibbald offered some tips for improving speed. "AutoCAD is pretty simple in its use of graphics: it draws lines, puts up raster text, and fills a polygon of up to ten sides. OS/2 drawing time is two to three times that of our direct-to-screen routines. The biggest bottleneck is in drawing lines. At first we did a GpiMove and a GpiLine for every line. That was very expensive. Then we started matching endpoints of lines. We buffered up to fifty lines whose endpoints matched, and then called GpiPolyLine and obtained a 10% performance improvement. After talking to Microsoft about the problem, they put a new call into OS/2 1.2, GpiPolyLineDisjoint. It works like GpiPolyLine, but the end points do not have to match. That gave us an additional 20% performance improvement."

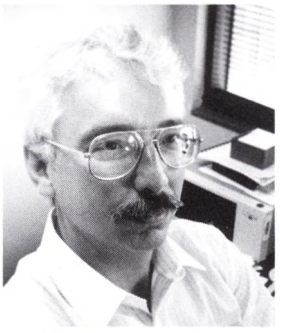
Sibbald commented that a second advantage of AutoCAD in the 640 KB DOS and Phar Lap environments is that it is a small data model, so there are no segment loads for accessing data. On OS/2, AutoCAD is a large data model. "The OS/2 compiler does not optimize things well. So we spent forty hours hand-tuning variables and putting in NEAR statements. That resulted in a 20–30% performance improvement. But there is only so much data you can move down to the near segment."

Gary Scott

Gary Scott described another tip about semaphores. "Both the ACAD thread and the PMACAD thread write to the presentation space. We needed a semaphore to keep the two from clashing. We first tried using DosSemSet and found the overhead slowed down the process by 10%. We reduced the overhead to 0.5% by implementing our own semaphore. The FAR call and return for a system semaphore cost about sixty clock cycles, which is about what we needed to provide our own semaphore. We normally would not use our own semaphores, but this is an area of code that handles many vectors per second to the display."

DYNAMIC DATA EXCHANGE

According to Phil Ford, Autodesk's objective is to make AutoCAD interact with as many programs as possible. Microsoft Excel is one of the programs with which AutoCAD currently interfaces. One use of this interface is to allow variables in the spreadsheet (such as part measurements) to control how the drawing is made in AutoCAD, as shown in Figure 3. The spreadsheet can calculate numerous parameters relating to a design, such as radius of gyration, volumes, moment of inertia, horsepower, torque, and so on. The user also can do what-if analysis on pricing from cost data supplied to the spreadsheet by AutoCAD. AutoCAD provides full access to its database in an ASCII form across the DDE link. Ford commented, "Today, it takes quite a bit of work to make DDE useful for users. There are two ways to interface with AutoCAD through the DDE code provided with the product. Source code is also provided. The first is



Phil Ford

with a LISP program and the second is through a specific ADS application, perhaps written in C."

For example, Microsoft designed the Excel product packaging using AutoCAD. With the DDE link to Excel, the spreadsheet calculated the cost of various alternative designs from data supplied directly from AutoCAD.

Today, AutoCAD does not support hot-links like Excel does. AutoCAD will not look at DDE information until the user requests that it be done. Hot-links are not always desirable. Phil Ford said, "The user may not want AutoCAD to attempt to redraw an image until all the new parameters are entered in the other program (such as a spreadsheet). Using spreadsheet terminology, you want to turn off automatic recalc in AutoCAD until all the changes are made in the spreadsheet."

OS/2 2.0

"The next OS/2 release of AutoCAD will be version 11, sometime during 1991," Ford continues. "We have not yet decided if it will be a 16-bit product (able to run on both OS/2 1.2 and 2.0) or if it will be a 32-bit application. OS/2 2.0 makes more sense to us. AutoCAD needs the speed and the flat memory model of OS/2 2.0. We will be asking the marketplace what they want before making the decision."

Steve Mighetto

Steve Mighetto described some of Autodesk's experiences with OS/2 2.0. "The first version of the Software Development Kit (SDK) had some problems for very large applications like AutoCAD. Our executable was over 2 MB, not counting the DLLs. IBM/Microsoft SDK Prerelease 2 compiles, links, and runs AutoCAD just fine. It appears that much of OS/2 2.0 PM has not yet been converted to 32-bit. Therefore, OS/2 2.0 graphics calls go through a thunk layer to convert them to 16-bit operations. That slows things down. Since AutoCAD is graphics intensive, our OS/2 2.0 version is about the same speed as the 1.2 version. We did a test by turning off all the graphics calls (AutoCAD only put raster text on the screen) and it ran at least twice as fast as the 1.2 version. This makes us feel that the compiler and linker are working well and that the performance degradation is being introduced by the thunk layer."

Kern Sibbald expects full 32-bit support in the next or the following release of the SDK. He said, "Within twelve to eighteen months of OS/2 2.0's shipping date, 50% of Autodesk sales will be AutoCAD on OS/2 2.0."

THE MOLE

The digitizer is the most common pointing device for AutoCAD users. Unfortunately, it does not function as a mouse in an OS/2 environment because it cannot get to the system menu bar. Users are forced to have both a digitizer and a mouse on their systems. The "mole" driver was created to allow the digitizer to operate like a mouse. The term "mole" was not chosen because it is another furry little animal running around on the desktop. Autodesk programmer Gary Scott chose the term because of its relationship to undercover espionage. The mole driver can function as a digitizer driver or a mouse driver. It fools the operating system into thinking that it works for it, when in reality, it is a double agent for AutoCAD. It takes the signals from the digitizer and feeds them into

OS/2 2.0 graphics calls go through a thunk layer to convert them to 16-bit operations.

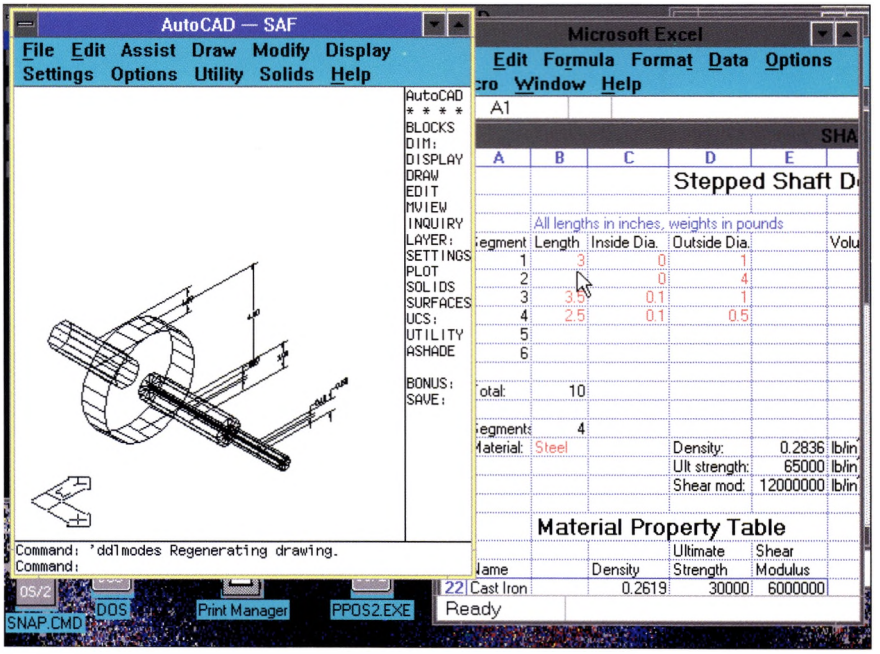


Figure 3. Excel Controlling the Part Drawn by AutoCAD

the operating system as if they came from a mouse. The digitizer can be toggled between digitizer mode and mole mode.

THE MARKET

Steve Mighetto describes the market for the OS/2 version of AutoCAD as follows. "AutoCAD for OS/2 is the best environment for those who write software for the eight-hour-a-day AutoCAD power user. It has all the tools, multitasking, and so on. The debugging tools are better than for any other environment. Our developers are not big companies that can afford big UNIX workstations. They have 80386-based machines already. All they need is OS/2 and AutoCAD and they can develop in a multitasking environment. During the first few months of sales, developers were the largest segment of sales."

Since AutoCAD is used mostly by power users, there would be little benefit to a Windows version because it would be slow.

Kern Sibbald added, "Users who want ease-of-use and added functionality, not pure speed, buy the OS/2 version. The OS/2 version adds new functions like DDE, the File Open dialog box, clipboard, new options on the Shell command, and dynamic reconfiguration of the graphics screen. To have truly integrated applications is very nice. In addition, OS/2 provides database connectivity that can be very useful with AutoCAD. Corporate users connected into a network appreciate these capabilities. OS/2 allows AutoCAD to communicate with other programs for the first time. The OS/2 version includes ADS, our C binding. No other platform has it. Early in the project, we felt that DDE was going to be an important feature of OS/2."

INTERNATIONAL VERSIONS

Autodesk produces international versions overseas: Swiss, German, French, Italian, Swedish, Danish, Norwegian, and Spanish. Autodesk U.K. has produced several Eastern European language versions. These have not required many changes to the program code.

Steve Mighetto said, "OS/2 National Language Support, where the program loads a DLL with the translated version information, looks very nice. AutoCAD does not use it because it was not available when we began the project." Sibbald added, "As time goes by, we will take advantage of more of those features."

OTHER QUESTIONS FOR THE DEVELOPERS

Developer: Do you have plans to do a Windows version of AutoCAD?

Sibbald: AutoCAD will not port to Windows and run efficiently. Since AutoCAD is used mostly by power users, there would be little benefit to a Windows version because it would be slow. A Windows version with the Phar Lap technology might be accepted. The ability to do that is not excluded with Windows; however, it is not a supported use of Windows. All the tools, such as compilers and linkers, are not there yet. It appears to us that Windows SDK will evolve to include 32-bit compilers.

Developer: From your extensive cross-platform experience, how does OS/2 stack up to others?

Sibbald: We love OS/2. It is much easier to program than UNIX. It is a little less mature, but it is a better operating system. It has many features that UNIX does not have and probably never will. Threads do not generally exist, and there is no way to do a timed wait of less than a second on UNIX. OS/2 DLLs get us around many problems. We can put something together in a day to experiment with, whereas it is much more difficult to do in UNIX as a separate process communicating through pipes. We can have separate parts of the program calling the same DLL using separate data areas. On the Macintosh, you would have to link separate code into each AutoCAD program that uses it, replicating code and increasing the size of programs. DLLs avoid symbol clashes too.

Scott: OS/2 is a wonderful development environment. It takes only 1.5 minutes to link AutoCAD, so we can make many changes during a day. We cut our link times in half by putting a PROTMODE statement in a DEF file. Multitasking allows me to do a compile and link in the background while I am editing a text file. That really speeds things up.

Developer: What sources did you rely upon for OS/2 training?

Sibbald: We gave the operating system to the programmers and said let's go!

Ford: Type QH (for Quick Help). Quick Help is great.

Mighetto: View in OS/2 2.0 is very good. The sample programs are very helpful.

Scott: The Gordon Letwin book[1] is very useful too.

Developer: What advice would you give to a small programming shop just getting started?

Scott: Don't throw a lot of programmers at it. Our four people were a good size group. More people will slow things down.

Mighetto: Start with a small sample program that comes close to what you are trying to build. Then expand from there. Take advantage of DLLs. You can develop a function working with a sample stub program and then plug it into the real application. When we first started, the user interface was running off a little program stub that had nothing to do with AutoCAD. From there, we were able to get much of the interface working. In parallel, others were working on internal aspects. Kern merged the two parts through the common memory buffer. That worked well.

Ford: Read Petzold's book[2]. Become familiar with Quick Help. This is one area where OS/2 far excels UNIX. The hypertext capabilities of Quick Help are far superior to man-pages. All the C calls, window calls, and all the tools and command line options for them are documented there.

Sibbald: If you do not have any experience with windowing program development, definitely take a class. At least one person on a team needs that experience. The hard part is redesigning your application. Be sure you have someone who is good at it. If you structure your application wrong, it will be even more difficult.

If you have never programmed with multi-threaded applications and semaphores, do not try to implement them in a production program until you are well grounded in the concepts. Once you understand them, they are very useful. Take the time to learn what DLLs are and how they work. They are very useful, but difficult to use in the beginning. DLLs cause you to separate code; it makes you think through the interface between modules.

Developer: Who should developers contact at Autodesk for support in interfacing to your products?

Sibbald: We have two programs that support third-party developers. The Registered Developer Program has been largely responsible for the 700+ applications that interface to AutoCAD. Another program, called the Strategic Developer Program supports the most sophisticated developers with products in key markets that we have identified. Anyone interested in getting information about either program should contact Joyce Williams at Autodesk at (415) 332-2344.

Autodesk

Founded:	1982
Employees:	1,000
Sales Channels:	3,000 dealers worldwide Direct sales force
First PC Product Shipment:	AutoCAD, December 1982
Other Products:	AutoSketch® AutoSolid® AutoShade® AutoFlix™

AUTODESK®

[1] Letwin, Gordon. *Inside OS/2*. Microsoft Press, 1988. 304p.

[2] Petzold, Charles. *Programming the OS/2 Presentation Manager: The Microsoft Guide to Writing Applications for OS/2's Graphical Windowing Environment*. Microsft Press, 1989. 864p.

Multimedia & Graphics

IBM Multimedia Product Chronology

Scott Swix

by Scott Swix

This article presents a chronology of IBM personal computer multimedia products over the past decade. During the past year IBM has announced three major hardware offerings—the IBM M-Motion Video Adapter/A, IBM Video Capture Adapter/A, and the IBM ActionMedia 750 series. A variety of IBM software products, such as the Audio Visual Connection® (AVC), LinkWay, Storyboard Plus, the InfoWindow Presentation System (IWPS), and Learning System/1 (LS/1), support these hardware offerings.

IBM multimedia products have evolved significantly since the early 1980s. Today, IBM has hardware to support complex presentation graphics systems, still- and full-motion video, digital audio, Musical Instrument Digital Interface (MIDI) music controls, and extensive interactive user input devices. Together these devices and software packages from IBM and its business partners provide a wide range of multimedia capabilities. The following chronology of some IBM products illustrates the backgrounds, differences, and directions of the present offerings. It is not intended to be an exhaustive list of features and products.

1981

PC Storyboard

The IBM PC Storyboard, the first IBM multimedia product for personal computers, enabled users to build slideshow presentations of graphic images. Operating with the four-color CGA display, it included three major components:

- A graphics Picture Maker/Editor that provided graphics and text functions to create graphic images for use in the story.
- A Story Editor that enabled the user to create sequences of images to display, the wipes and transitions between images, and the user action or time delay to cause progression to the next image.
- A Story Teller that played back stories created in the Story Editor.

These logical components are appearing today in many subsequent products from IBM and other vendors.

1984–1986

Toward the mid-1980s, IBM improved its graphics systems with EGA and added new solutions to the selection of multimedia products, including integrated audio, video, and touch.

InfoWindow Touch Display and Control Program

The IBM InfoWindow Touch display also was introduced during this time period. This was IBM's strongest entry yet in the field of integrated multimedia. The InfoWindow display, combined with a personal computer, offered EGA graphics, digital speech and sound, and a touch-sensitive screen. In addition, audio and video sources could be connected. The InfoWindow could play stereo audio through either internal or external

speakers, and full-motion video from any NTSC video source. The graphics and video can be combined on the screen to display either or both, on a pel-by-pel basis. These features have made the InfoWindow an industry standard for interactive video systems. Applications include medical education, substance abuse presentations, storefront marketing and merchandising, repair and maintenance education, public access for government services, and drivers' education and license testing.

The InfoWindow Control Program that provides access to the video, audio, and touch functions can be used to program the InfoWindow directly. A variety of business partners have developed authoring systems based on the InfoWindow Control Program. There are now more than 20 development packages and over 600 applications.

Two IBM authoring systems, InfoWindow Presentation System (IWPS) and Learning System/1 (LS/1), are offered for use with the InfoWindow. Both help users in designing, developing, and implementing multimedia presentations.

InfoWindow Presentation System

IWPS is an authoring and presentation system for IBM InfoWindow. Its primary use is in the merchandising, marketing, and public access environments. It is designed as a timeline, where table entries control the display of graphic images, video, input/output, and text generation. Input data and decision logic can generate actions and branching to other timelines. There is also access to external programs, such as custom code and communications.

Learning System/1

LS/1 is an education- and training-oriented authoring and presentation system for IBM InfoWindow. It is built on the educational paradigm, providing templates for questions, menus, tests, and presentations. Extensive answer judging permits evaluation of multiple answer types. Student tracking and scoring functions enable group and individual performance analysis, student progress, question pools, and so on. Integrated presentation editors provide control of video, synthesized speech, textual windows, and both picture and vector graphics. LS/1 includes an extensive language to permit custom operations and detailed control of the application.

Figure 1. IBM InfoWindow Touch Display

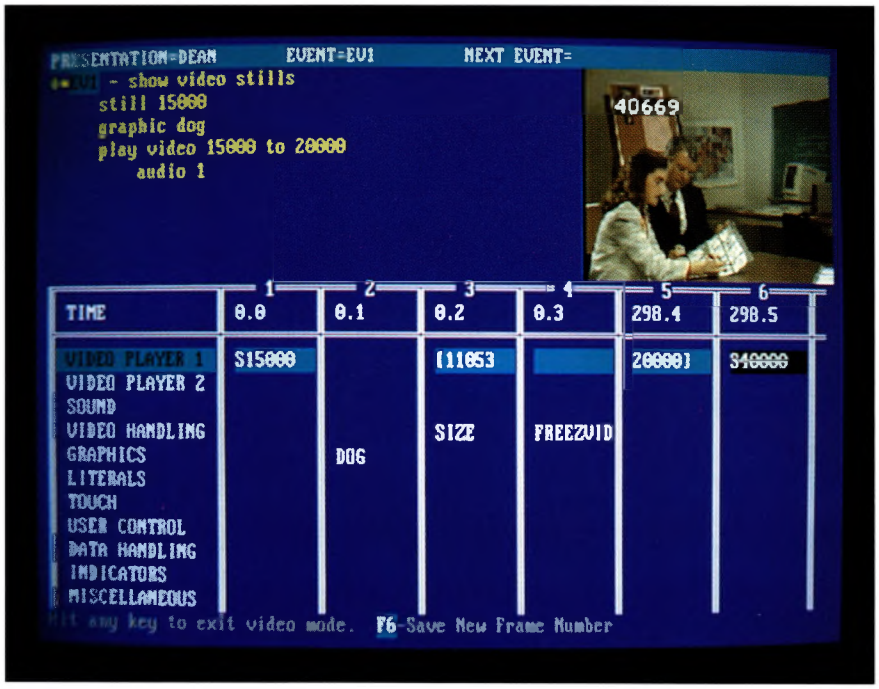


Figure 2. Sample IWPS Screen

1987

IBM Music Feature Card

The IBM Music Feature Card is IBM's first entry into the field of audio control and generation. It supplied the user with musical programming via MIDI, a standard in the music industry to control music devices and generate musical notes and scores. The Music Feature Card has FM stereo sound, 336 preset and programmable voices/instruments, and 8 simultaneous voices for ensemble performances. It provides a means to create computer-controlled and edited scores, songs, and special effects.

PS/2

As the decade continued, IBM redesigned its microcomputers and announced the IBM PS/2 line in 1987. The PS/2 series integrated new functions, including a new graphics adapter, the VGA. The VGA provides enhanced resolution, up to 640x480, and the improved color palette allows users to select any 256 colors from a palette of 256,000. Further graphics capabilities were made possible with the optional 8514A graphics adapter, supporting 256 of 256,000 colors at a 1024x768 resolution, and a variety of hardware graphics functions.

Figure 3. Sample LS/1 Screen

With the new hardware, IBM also announced enhancements to some of the existing products. These included an adapter for the InfoWindow to operate with a PS/2, and a new version of Storyboard—IBM Storyboard Plus.

Storyboard Plus

Storyboard Plus, built on the original Storyboard product concept, has many improvements. Storyboard Plus was redesigned to take advantage of the VGA, offering enhanced color and resolution modes. Some major additions are a new graphics-menu user interface, support for video digitizing boards, expanded graphics paint functions, more powerful story editing, and control and image manipulation functions. Other new features include some animation capabilities, support for the IBM Music Feature Card, and libraries of existing images for users' convenience.

1988

In the late 1980s, IBM announced a new selection of both hardware and software platforms. These offerings have significantly expanded the range of multimedia solutions available. One of the most important of these is OS/2, which provides a powerful base for creating multimedia tools and applications. By supporting multitasking, interprocess communications, expanded program memory space, and the Presentation Manager (PM) user interface, OS/2 enables a new level of multimedia systems to be developed.

IBM Multimedia Shell Utility

Development for the InfoWindow continued with the announcement of the IBM Multimedia Shell Utility. This product, designed for public access applications, provides a simple method for creating menu-based interactive video presentations.

IBM LinkWay

LinkWay, an integrated editing and presentation tool for creating hypermedia applications, was announced during the same period. Designed for the K-12 education market, LinkWay includes a graphics picture editor and a presentation language, as well

as the ability to create hyperlinks between words or objects on screen and actions or other data items in the presentation. LinkWay operates on the full range of IBM PC and PS/2 computers. It provides support for IBM InfoWindow so that presentations can include motion video.

1989

Video Capture Adapter/A (VCA)

The VCA board is a Micro Channel real-time frame grabber. It captures still images of the video coming in from NTSC, Y/C, or RGB sources, and can save the data to a file. The VCA can freeze-frame full-motion video to obtain a still image in 16 bits, or 64,000 colors. Software can convert the 64,000-color picture to a VGA 16- or 256-color image.

The board is required for capturing the images only. The saved images can be shown on other systems that do not have the VCA. Although each picture is only a still image, users can display many still images sequentially for a form of motion. This technique normally is restricted to small areas of the screen.

The main use for this card is capturing photographic quality images for presentations, catalogs, training, and other applications that require high-quality still pictures.

IBM Audio Capture and Playback Adapter (ACPA)

The ACPA card provides real-time digital audio capture and playback for both the Micro Channel and PC-XT/AT bus computers. One of multiple audio inputs can be chosen and then digitally stored to system memory or disk. The resulting audio file can be edited alone or with other audio files and live audio for mixing applications. The ACPA card offers a variety of audio compression levels and qualities, allowing users to save the audio in voice, music, or stereo fidelity.

Once the audio file has been captured and edited, if desired, the audio can be played back through the ACPA board for use in multimedia or simple audio applications. Sample uses include audio dubbing over video, audio accompaniment with images, sound effects, multilanguage support, music history and study, and so on. IBM recently announced a lower cost version of the ACPA adapter and added compatibility with the IBM Speech Adapter.

Audio Visual Connection (AVC)

AVC is a software product that integrates images, digital audio, media production, authoring, and presentation capabilities. It works with the Video Capture Adapter and the Audio Capture and Playback Adapter. AVC operates in DOS and OS/2, and connects to IBM's host-based KnowledgeTool® Expert System. It provides a powerful platform for developing multimedia applications. Figure 4 illustrates the various multimedia components that can be used to create and deliver presentations using AVC.

AVC integrates images, digital audio, media production, authoring, and presentation capabilities.

Since it is designed with a spreadsheet-style interface, table entries and selections control image display and sequences, audio synchronizing, special effects, data entry and output, and text display. Hypertext-like linkages can be built into screen images. AVC also possesses an extensive authoring language called AVA that includes I/O, field operations, variables and calculations, decision logic, and external program calls (see "Programming the Audio Visual Connection" in this issue).

Storyboard Plus 2.0

In 1989, IBM upgraded Storyboard Plus to provide higher resolution graphics, more graphics art, and story control functions to support more devices, and to permit use of the Video Capture Adapter/A for digitizing images into the Picture Maker program. Storyboard Plus 2.0 supports the IBM Audio Capture and Playback Adapter, thereby providing new audio capabilities.

1990

At the beginning of 1990, IBM increased the pace of multimedia product introductions. This year IBM announced two new motion video products and upgrades to existing software solutions.

The M-Control Program is a set of libraries and tools for building applications.

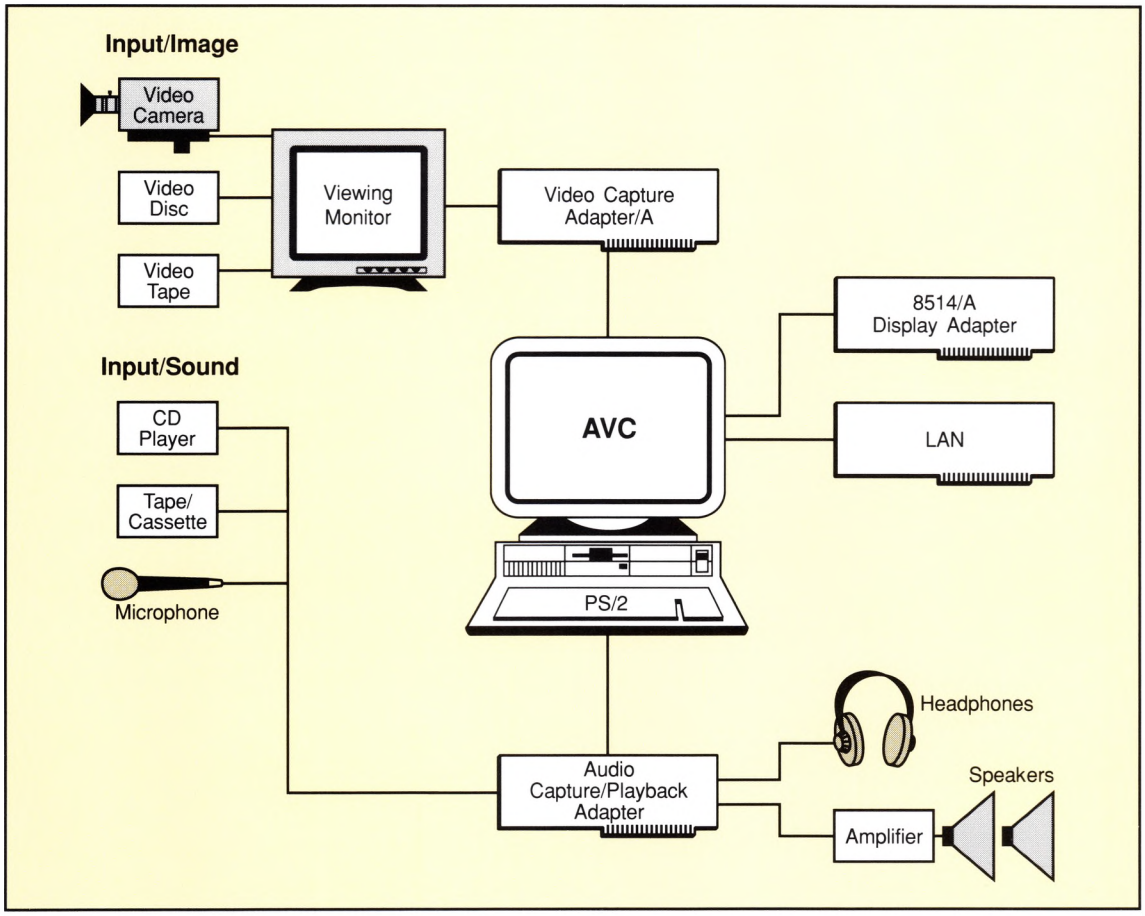


Figure 4. AVC Attachment Options

IBM M-Motion Video Adapter/A and M-Control Program

The M-Motion Video Adapter/A, with the M-Control Program, provides both still- and full-motion video on PS/2 monitors. Since it is a compatible extension of the InfoWindow video capabilities, this new hardware and software combination supports over 600 applications and 20 authoring systems written for the InfoWindow. Operating on Micro Channel computers, the M-Motion Video Adapter/A also provides new functions.

The M-Control Program contains device drivers and programmers' toolkits for the DOS and OS/2 PM environments. The M-Control Program is not an application. It is a set of libraries and tools for building applications that can include a variety of video and graphics manipulations available on the M-Motion Video Adapter/A.

Some video functions supported are multiple NTSC, S-VHS, or PAL sources that can be connected and controlled simultaneously. Software can manipulate video source selections, full-screen and area video freeze, color, tint, brightness, and still video image save/restore. All video is displayed in 21-bit YUV, for a total of two million simultaneous colors.

Any region of the incoming video can be shown full-size or scaled and placed at any location on the screen. This allows motion video to be displayed in OS/2 PM windows, simultaneously and integrated with other applications, such as spreadsheets and word processors. Support for libraries and controls for Microsoft Windows 3.0 has recently been added in version 1.1. The new Windows programming interface is based on the existing OS/2 PM interface, which simplifies the development of applications that can run in either environment.

Video and graphics are combined at the pel level via controllable graphics overlays in all VGA modes. Video is shown in the background. VGA graphics are displayed on top

of the video wherever desired. Therefore, applications can show only video, only graphics, or any combination of graphics and video.

The M-Motion Video Adapter/A possesses the capability to play and control two line audio sources. The audio can be digitized, that is, saved to disk and replayed later. This permits voice-overs, multilingual applications, and so on.

Audio Visual Connection 1.03

A new version of AVC was recently announced that supports the integration of motion video via the M-Motion Video Adapter/A, adding a new media for use in AVC presentations. Support for playing MIDI music files also has been added.

ActionMedia 750 Series

The ActionMedia 750 adapters for Micro Channel computers, jointly created by IBM and Intel, represent a new developmental technology from IBM called Digital Video Interactive (DVI). The ActionMedia Capture Adapter and Delivery Adapter can receive and display incoming motion video and audio, modify and digitize both in real-time, compress and store the result to disk, and later replay the video and audio from disk.

The DVI approach eliminates the need to store and distribute video exclusively on videotape, videodisc, or other video-specific media. It allows live video to be captured for modification or replay. Resulting video and audio can be stored effectively on CD-ROM, transmitted over local area networks, and otherwise treated as normal data files. Note that even with compression, digitized real-time video is disk intensive. It is best to use large-capacity hard disks, CD-ROM, or network drives.

Some features of the ActionMedia 750 products include full-color, full-screen, 30-frame-per-second playback from hard disk or CD-ROM. Other capabilities are video special effects, 24-bit color, two audio inputs, and audio mixing. The software supplied with the ActionMedia products supports still- and motion-video capture, audio capture, audio and video editing—all packaged in a set of C language runtime libraries.

The ActionMedia products and software are development tools, not applications. Developers can design and implement applications using the new ActionMedia system as the audio and video control system.

Figure 5. Sample Storyboard Plus Screen

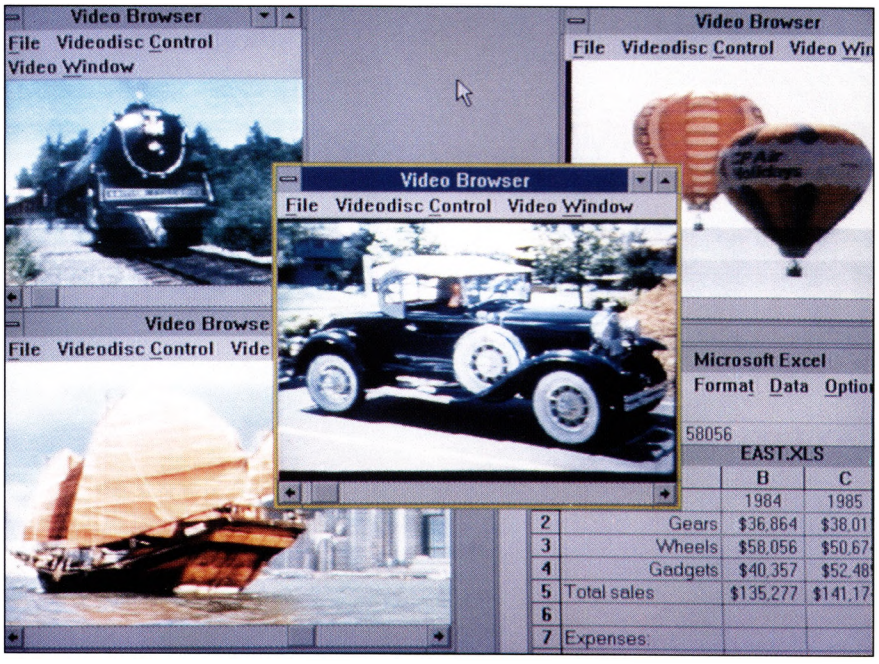


Figure 6. Sample M-Motion Control Program Screen

CD-ROM

The IBM CD-ROM drive, announced in 1990, provides a new method for distributing multimedia courses and presentations. The capacity of CD-ROM discs, as well as the simplicity of use and mass production, greatly reduces the problems with distribution of normal data, long training programs, DVI motion-video sequences, and digital or analog audio.

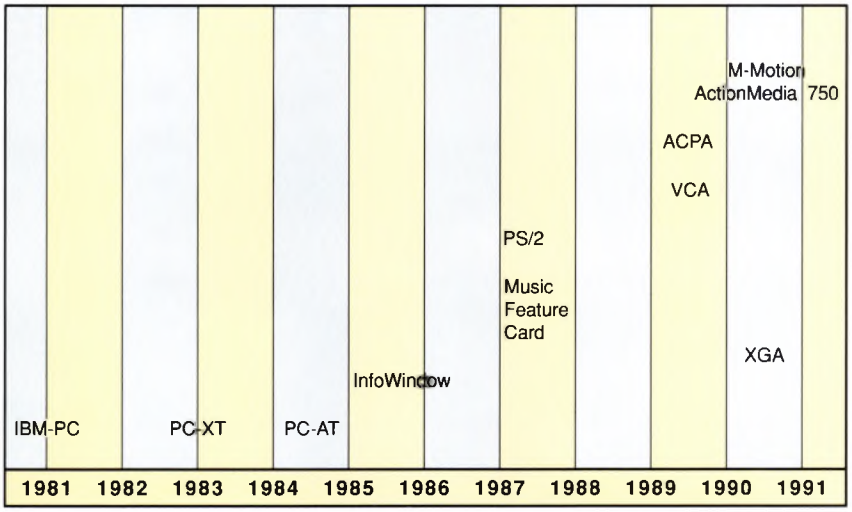


Figure 7. IBM Multimedia Hardware Timeline

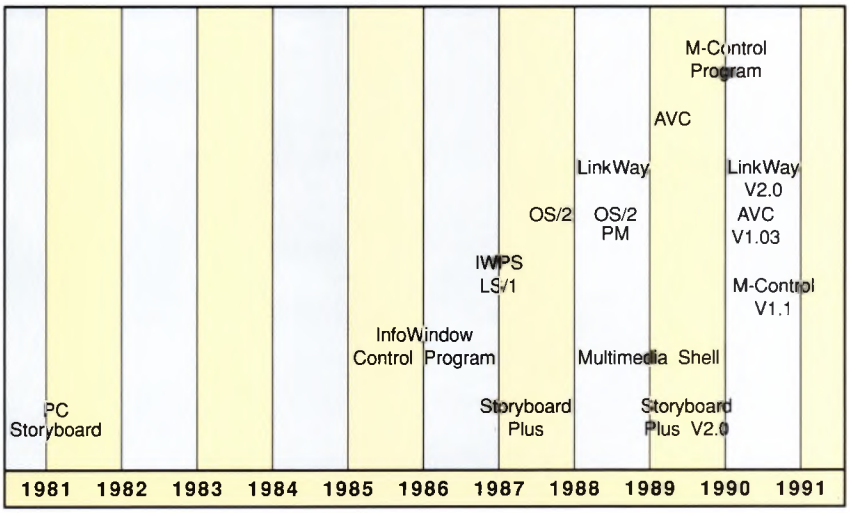


Figure 8. IBM Multimedia Software Timeline

LinkWay 2.0

In July 1990, IBM announced LinkWay version 2.0, which enhances all areas of the product. It now supports IBM M-Motion Video Adapter/A and IBM Audio Capture and Playback Adapter to provide expanded motion video and digital audio capabilities. Other new features include a high-function script language, contextual helps, improved graphics paint and art functions, and scrollable hypertext. Enhancements include graphics resolutions, printer support, and performance.

XGA Graphics Adapter/A

The XGA adapter is the new high-end IBM graphics adapter. Providing 8514A and VGA compatibility, it runs ten times faster than the 8514A and acts as a Micro Channel bus-master adapter, which further improves performance and access to system memory independent of the system CPU. Additional features include hardware bit-blitter and cursor functions and up to 1 MB of on-board video RAM. This provides 1024x768 resolution with 256 colors.

Figures 7 and 8 summarize the chronology of IBM multimedia software and hardware products.

Scott Swix, *IBM Entry Systems Division, 4111 Northside Parkway, Atlanta, GA 30327. Mr. Swix is a staff programmer in Multimedia Product Development. He joined IBM in 1984 and has been involved in the design and development of various multimedia products based on the IBM InfoWindow and IBM M-Motion Video Adapter/A. He holds a BS and is pursuing an MS in Information and Computer Science from Georgia Institute of Technology (Georgia Tech). Mr. Swix also teaches Computer Graphics at Georgia Tech as an adjunct faculty member.*

Multimedia & Graphics

Programming the IBM M-Motion Video Adapter/A and M-Control Program

by Scott Swix

This article discusses the features and programming of the M-Motion Video Adapter/A and M-Control Program in OS/2 Presentation Manager (PM). This new adapter and software package provides advanced still- and full-motion video capabilities, multiple video source control, stereo audio, and VGA graphics overlay in OS/2 PM and DOS environments.

IBM recently announced two new multimedia products, the M-Motion Video Adapter/A and M-Control Program, which bring previously unavailable capabilities to OS/2 PM applications. Together, these products permit programs to include high-quality still video, motion video, and stereo audio.

The M-Motion Video Adapter/A is a Micro Channel video board that expands on the video functions of the IBM InfoWindow Touch Display. The M-Control Program has a programmer's toolkit and selection of device drivers for both the DOS and OS/2 PM environments. The M-Control Program is not an application, but a base for building applications. It uses the M-Motion Video Adapter/A, which can perform a variety of video, graphics, and audio manipulations.

The M-Control Program OS/2 PM Toolkit features the following capabilities:

- **Multiple NTSC, S-VHS, or PAL Sources**: Three inputs can be controlled simultaneously. These devices include videodisc players, VCRs, video cameras, TV tuners, and so on.
- **Multiple Stereo Audio Inputs**: Several audio inputs can be attached and controlled separately. The audio can originate from the same device as the video (as in a videodisc player) or from a separate source.
- **Enhanced Video Manipulation**: Any incoming video region can be shown full-size or reduced, and placed at any location on the screen. All video is displayed in 21-bit YUV, for a total of two million simultaneous colors.
- **Software Controls**: Software can control video and audio source selection, color, tint, brightness, volume, bass, balance, and treble. Software provides video image manipulations, such as full-screen and area video freeze and protect. It also can save and restore variable-size still video images or create a bitmap approximation of a video image.
- **Combined Video**: VGA graphics and the video being shown can be combined by a pel-level controllable graphics overlay in all VGA modes. This enables applications to display only video or graphics, or any combination of graphics and video.
- **Two New OS/2 PM Window Classes**: Provided to simplify programming.
  - The Video Window Control class projects video in OS/2 PM windows, simultaneous and integrated with other applications that can include spreadsheets and word processors.
  - The Videodisc Player class communicates with many third-party videodisc players to control the position, motion, and other attributes of the player.

The M-Control Multimedia Toolkit for OS/2 PM presents a new approach to integrating multimedia. It uses object-oriented control classes that simplify the control of the devices and presentation of the media. Figure 1 shows two full-motion windows displayed

Scott Swix

These products permit OS/2 programs to include high-quality still video, motion video, and stereo audio.

by the M-Motion Control Program video browser.

PROGRAMMING MULTIMEDIA APPLICATIONS

Some basic technical considerations apply to any application or program designed to use motion video. The most important of these include:

- How to show video on the screen
- How to control the video source device

A computer, display, video source, and a video adapter, such as the M-Motion Video Adapter/A, can integrate these components. The next issue is coordinating the video to be displayed, determining where it appears on the screen, sizing its image, and choosing its attributes, such as tint, color, contrast, and so on.

The intent in OS/2 PM is to display the video in a window. The requirements for creating and controlling windows include defining the class, attributes, and window procedure. A video window should address these items. In addition, required special routines control the M-Motion Video Adapter/A to set the video inputs, region of the incoming source video to display, screen location for the video (within the window boundaries), transparency color, video scaling factor, and other related settings. Figure 2 illustrates some hardware and software manipulation required for displaying motion video in a PM window.

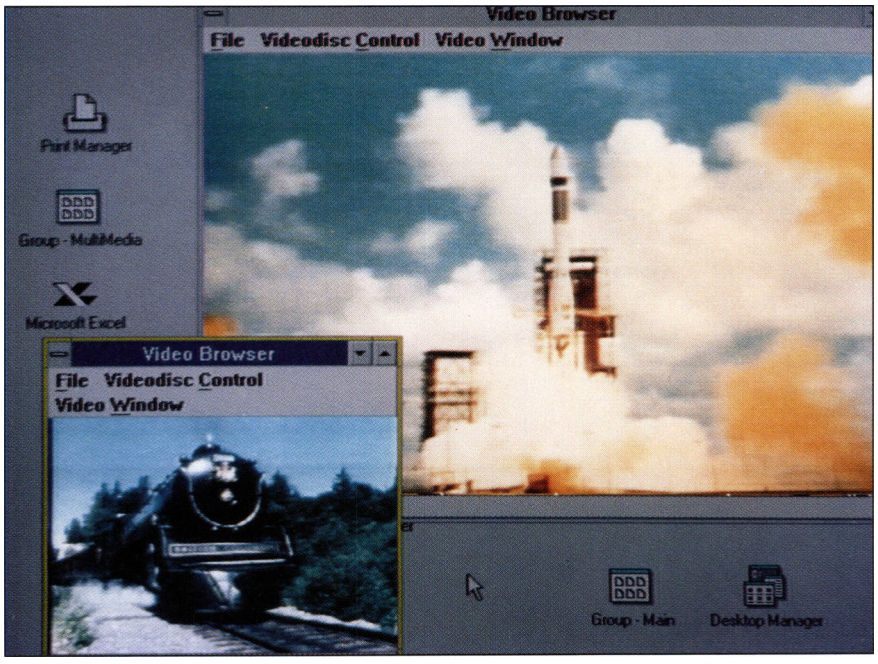


Figure 1. Full-Motion Video in a PM Window on a PS/2

The procedure for correctly manipulating these elements is complex in single-tasking, single window systems. The complexity rises quickly when considering multitasking and multiwindowed environments, such as OS/2 PM. The underlying hardware is not concerned with individual windows and their boundaries, nor with operations that change the size and position of the windows. The software must accommodate these situations to ensure that the video is displayed only within the window regions desired, and that the manifestation of the video is altered, as necessary, for changes in window size and location.

Video Window Control Class

The M-Control Program supplies a new customized Multimedia Video Window Control class, WC_MVWN. It handles all the common PM window requirements and those situations specific to showing video in an OS/2 PM window. This relieves the program from the mundane, but difficult, details of managing video in a window. The improvements in device independence eliminate the need for a program written specifically for the M-Motion Video Adapter/A.

The WC_MVWN class accepts all messages common to normal OS/2 windows, including WM_MOVE, WM_SIZE, WM_PAINT, and so on. It also accepts messages for controlling and querying various video attributes, such as color and brightness. The video within the window is maintained as the window is moved, sized, hidden, and redisplayed.

Finally, the WC_MVWN class provides some control over audio that may be associated with the video (for example, from a videodisc player). It supports many line-audio control messages to turn the sound on/off, change volume, balance, and so on.

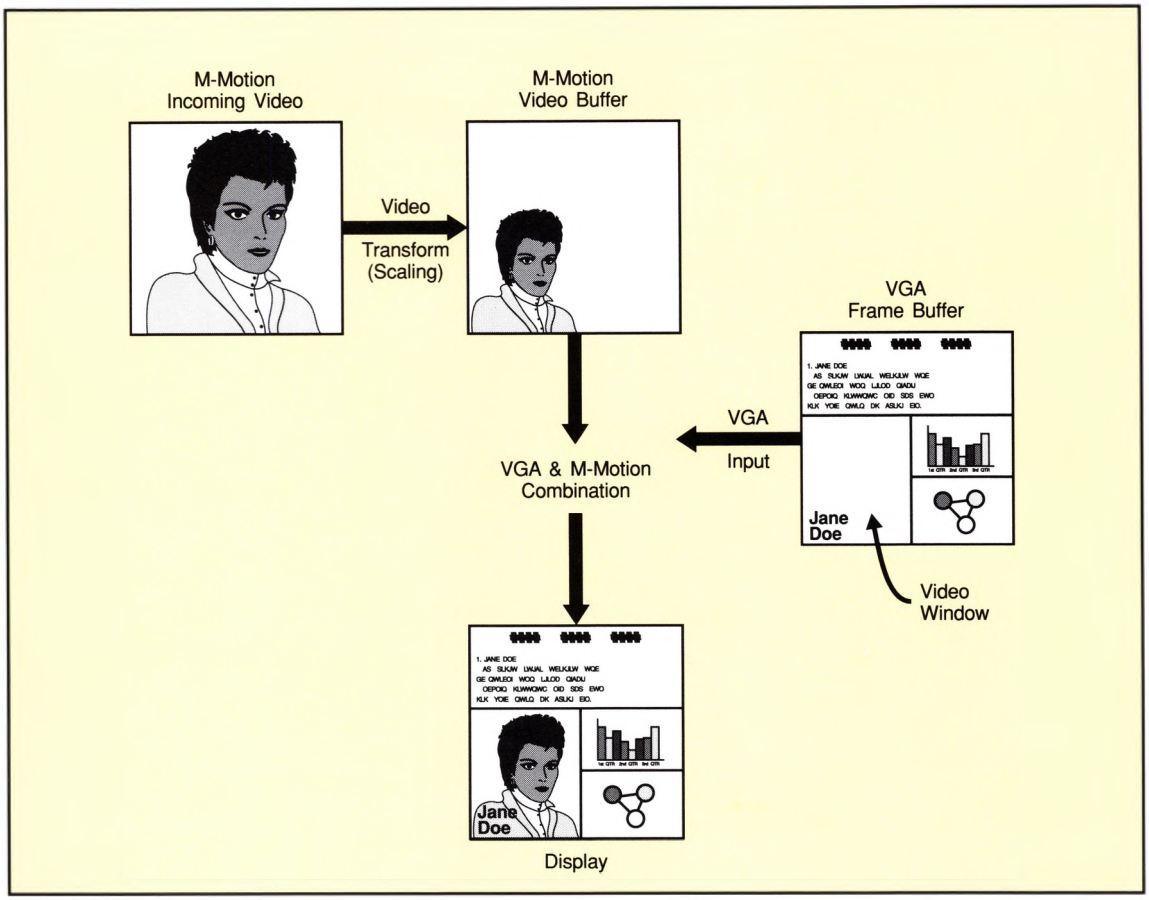


Figure 2. Combining Video and Graphics

The Toolkit performs the low-level communications and coordinates everything through OS/2.

Subclassing—Customizing the Video Window

A program should control the video window, so it can display graphics and process user input or other actions. By subclassing the video window, the program intercepts messages and adds additional processing, for example, displaying bitmapped graphics over the video. The result is a fully customized window that shows video. The program also may request a video window and never perform any custom processing. This leaves all message handling to the MVWN defaults.

Figure 3 illustrates the interaction of the video window with the other components.

Controlling the Video Source Device

Controlling a videodisc player or other video device is a complex task for a normal program. Since video can be shown in a window, how can the video source device be controlled?

Most videodisc players and other media devices that can be connected to a computer accept commands such as stop, play, seek to a location, change speed, and so on. These commands are built as data structures and sent over serial or parallel communication lines. The command may have to wait until the device is ready, and the system may have to listen for responses from the device. These operations must be strictly timed and ordered. Different protocols and requirements for each device, and even different videodisc players, create an additional complication. Consider also that more than one application may be attempting to control the device, requiring some form of arbitration.

The Toolkit performs the low-level communications and coordinates everything through OS/2 as necessary. The videodisc device drivers in the Toolkit that are responsible for communicating with the videodisc player

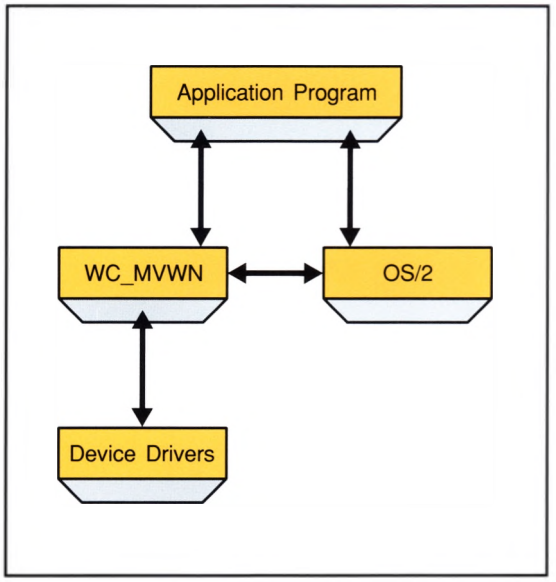


Figure 3. Video Window Interactions

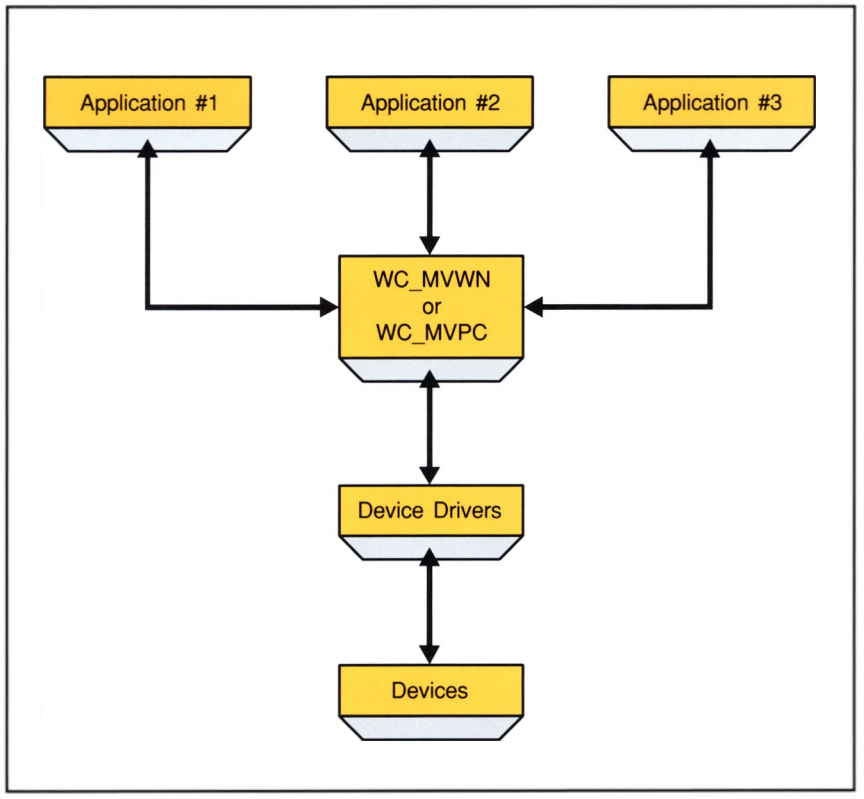


Figure 4. M-Control Resource Contentions

provide this support. WC_MVPC, the special Multimedia Videodisc Player Control class, handles communications with the device drivers and enables the application program to control frame position, motion, audio, speed, and other player attributes.

Resource Contentions

There may be two tasks, or one program with two controls, that attempt to access the same resource device (for example, the videodisc player). Therefore, an arbitration scheme has been implemented to control actions by the Toolkit objects, as shown in Figure 4.

Only one videodisc player control can communicate with the videodisc player at a time. Other MVPC objects can request services, but these are delayed until the current MVPC object relinquishes control. This scheme is based on a token, or permission flag, that is owned by the process currently communicating with the player. The process initiates a token request and transfer whenever a control attempts to communicate with a device, but does not have the token. This is transparent to the user and the application program.

Similarly, only one video window can show motion video at a time. All other video windows are considered inactive. The same token scheme serializes access to the M-Motion board for video display operations.

Operating the Controls

The first concern of application programmers is generating player actions, not dealing with the underlying details of how to accomplish this. Secondly, the user of the application may need a way to control the device directly, rather than perform all media operations in the code. Therefore, applications fall into two categories illustrating different levels of device control:

- **Applications without a User Interface**: The application must have direct control of the device without user interaction. All access to the media devices is through software.

- **Applications with a User Interface**: The application must allow the user of the program to have some control over the device. The developer, therefore, must provide a user interface.

For applications without a user interface, there must be the capability to request that a logical operation be performed, such as "Stop." The details, however, should be relegated to the Toolkit. The program must specify the logical operations as required by the application. The WC_MVPC control is responsible for the execution of these logical operations. Messages, such as which player is requested, what position to move to, where to begin or end a Play From-To sequence, are sent to the WC_MVPC control. It formats and passes these commands to the device drivers. The control provides response, acknowledge, and advise messages about the result of the requests. In addition, the class can be queried about the current status of the player.

The developer must supply a video control panel, scroll bar, on/off switch, or other interface available to the user for supporting a user interface. The WC_MVPC class can attach a Custom User Interface (CUI) to handle all user selections and related MVPC commands and messages. This simplifies the program in two ways. First, the CUI resides in a Dynamic Link Library (DLL) supplied by the device manufacturer or another source. Second, the CUI is "attached" to the videodisc player control when the control is initialized. From that point, the program is not required to perform any videodisc control operations. Subclassing and advise messages, however, direct and monitor the actions of the CUI and control. The Toolkit includes a sample scroll bar user interface.

Figure 5 illustrates the general interaction of the components.

Communicating with Controls

The program communicates with the controls through functions and messages. Functions are instantaneous operations, often related to the creation and destruction of objects and data structures. The WinPostMsg() function dispatches messages consisting of operational instructions and parameters between the control and the application.

The process of creating and communicating with a control follows this sequence:

1) Call the control's initialization function. This registers the class with the operating system and loads the necessary DLL. This function also assigns a set of message IDs to coordinate later communications.

2) Use the PM WinCreateWindow() function to create an instance of the control for use.

3) Begin sending messages to the control. This involves three steps:

   a) Call the control's CreateCmd() or CreateQry() function to obtain a structure to hold the message and parameters. The command structure differs from the query structure.

   b) Fill in the parameters for the message, and send the message using the PM WinPostMsg() function.

   c) The control interprets the message, as necessary, for the device and attempts

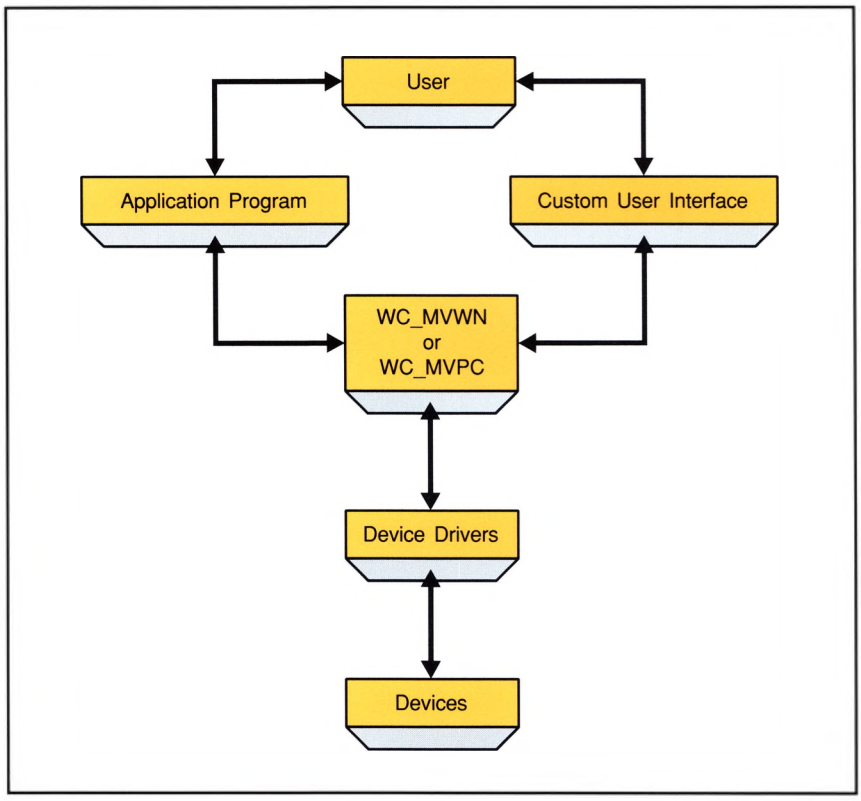


Figure 5. M-Control Communications

to fulfill the request. It sends an acknowledgement. A response may be transmitted later if the action is not instantaneous.

Wait for any acknowledgement and/or response. After all responses have been received, call the control's DestroyCmd() or DestroyQry() function, as appropriate. These functions deallocate the structure obtained by the Create functions (in step a, on the previous page).

4) When the program no longer requires the control, destroy the control using the PM WinDestroyWindow() function.

5) Finally, call the appropriate Terminate() function to release the instance of the control and the DLL.

Messages

The messages with the MVWN and MVPC controls include a wide range of device actions, queries, and advise commands.

Action messages cause the device to perform an action. A small selection of these for the video player is shown in Figure 6.

Similarly, some action messages for the video window are shown in Figure 7.

Queries are designed to obtain status information about the control and device, such as current position, display attributes, audio attributes, and so on.

Advises ask the control and/or device to inform the application when certain situations or actions occur. Several advise messages are shown in Figure 8.

Functions

A variety of other functions offer the ability to determine the system configuration, identify whether players exist and what type they may be, and create and destroy less commonly used data structures.

MVPC_CMD_STILLFRAME	Position the player at a frame on the videodisc.
MVPC_CMD_PLAYFROMTO	Play the current disc starting at the From frame and proceeding to the To frame.
MVPC_CMD_STOP	Cancel an outstanding play.
MVPC_CMD_AUDIO	Enable or disable the videodisc audio.
MVPC_CMD_MATCHFRAME	Set a frame to be monitored. When that frame is reached, notify the application.

Figure 6. Action Messages for the Video Player

MVWN_CMD_BRIGHTNESS	Set the output brightness to a new level.
MVWN_CMD_BASS	Set the bass tone level for audio.
MVWN_CMD_COLOR	Adjust the color of the video being displayed.
MVWN_CMD_CAPTUREIMAGE	Save a copy of the video buffer to memory.
MVWN_CMD_FADE	Fade the video toward/away from black.
MVWN_CMD_FREEZE	Freeze the current motion video image on screen.
MVWN_CMD_SCALE	Change the size of the incoming video being displayed by compressing a video source region into the video window.

Figure 7. Action Messages for the Video Window

MVWN_CMD_ADVISEACTIVE	Inform the application when this instance of the control gains the active token.
MVPC_CMD_ADVISESEEK	Inform the application when the player performs a seek operation.
MVPC_CMD_ADVISEFRAME	Inform the application when the player changes frame position and what that position is.

Figure 8. Advise Messages

CONCLUSION

The intent of this new methodology is to provide a simpler, faster, and more device-independent approach to controlling multimedia devices. The application programmer can be more productive *without* concerns about device-specific communications, multitasking resource contention, and other burdensome requirements.

We hope to expand this paradigm's application to other environments and extend the device-independent classes and messaging structures to new device types. This approach to enabling multimedia will allow developers to create applications for multiple systems, using logically similar, but mechanically different, media devices with much less effort and greater portability of code.

Scott Swix, *IBM Entry Systems Division, 4111 Northside Parkway, Atlanta, GA 30327. Mr. Swix is a staff programmer in Multimedia Product Development. He joined IBM in 1984 and has been involved in the design and development of various multimedia products based on the IBM InfoWindow and IBM M-Motion Video Adapter/A. He holds a BS and is pursuing an MS in Information and Computer Science from Georgia Institute of Technology (Georgia Tech). Mr. Swix also teaches Computer Graphics at Georgia Tech as an adjunct faculty member.*

Multimedia & Graphics

Spatial Analysis

Giulio Maffini

by Giulio Maffini

Geographic Information Systems (GIS) is a rapidly growing technology that promises to have a major impact on the computing environment of the 1990s. This article discusses the basic principles of GIS, how it is being used, and how a leading developer in the industry has used OS/2 to bring this exciting application to all users.

The acronym GIS will be appearing more frequently during the 1990s. The fundamental principle of GIS is that data is coded, organized, analyzed, and modeled through explicit reference to its spatial location and other geographic properties. GIS represents the world using geographic building blocks or entities such as points, lines, areas, surfaces, and networks as shown in Figure 1.

Points: Features associated with a single location. Examples of points are weather stations, lookout towers, wells, and so on.

Lines: Locations sharing the same characteristic. Rivers, roads, pipelines, and property boundaries are examples of lines.

Areas: Spaces contained within a boundary in two-dimensional space. Examples of areas include land parcels, forest stands, counties, or parks.

Surfaces: Three-dimensional spatial entities. At any point or location, the surface identifies some measurement or value that can be quantitative or qualitative. Surfaces can be used, for example, to represent population density, elevation, and temperature.

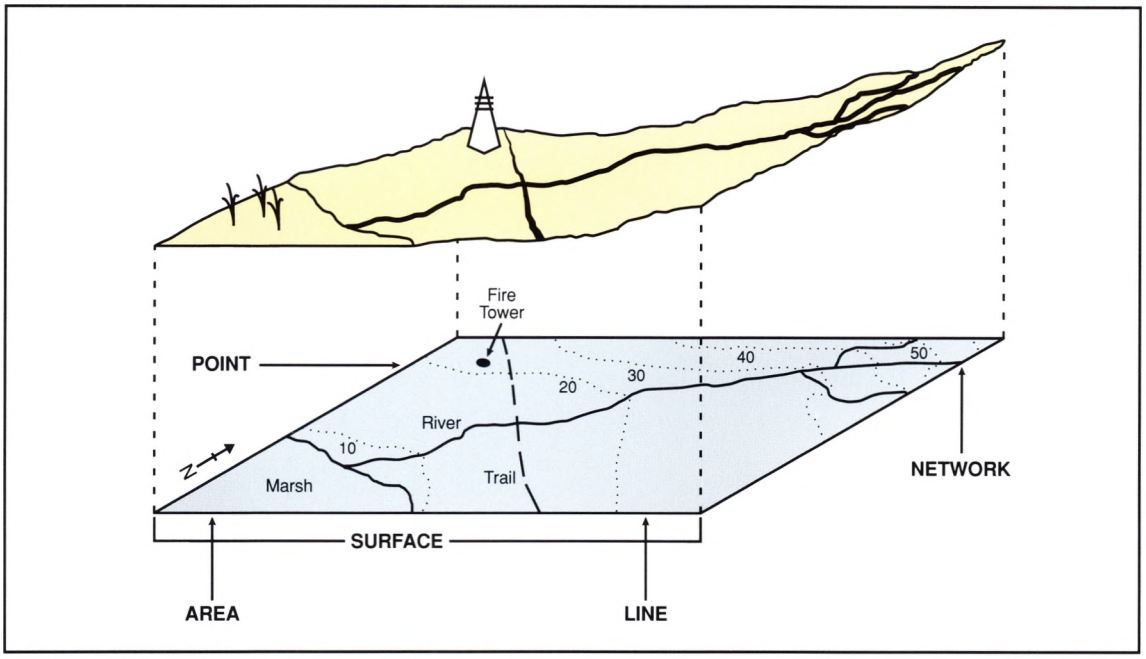


Figure 1. GIS Entity Types

Networks: A series of interconnecting lines and points along which there is a flow of information.

GIS is important to management and developers because much of the information about our natural and human world lends itself to being managed by geographic criteria. GIS can tackle problems that defy the capabilities of more conventional database management methods. For example, it is difficult for a conventional database system to answer the simple question "Where are all my customers within half a mile of this (arbitrarily selected) intersection?" Working with relational Database Management Systems (DBMS), GIS acts as a catalyst and produces new information processing technology of great power and flexibility.

The inspiration for GIS developed as cartographers began to look to computers for help. Their priority was to make the map drawing process more efficient and effective. Computer technology was used first to replicate manual processes. Cartographers used an instrument called a *digitizer* instead of drawing a line on a piece of paper. They created those same linear features as a digital record that could be stored, displayed, and edited in a computing environment. The discipline imposed by the computer technology for coding and displaying information created a great opportunity.

Others then began to use computers to analyze the information used to make maps instead of just making maps. At first they asked simple questions, such as, "How long are the arterial roads in Montgomery County?" or "How many schools are in Montgomery County and where are they located?" These simple requests retrieved, displayed, measured, or counted features that had been coded by the cartographers in the map making process. The appetite for more complex queries grew rapidly, generating questions such as, "Where are the wooded areas that are within two miles of the arterial roads?" or "How many people live within half a mile of each school?" The types of questions that we now can ask have reached a very sophisticated level.

SPANS OVERVIEW

Spatial analysis, now a discipline with a long and well-established theoretical base, is a powerful tool. SPANS®, a geographic information system developed by TYDAC Technologies, is an acronym for SPatial ANalysis System.

The GIS industry consists of companies that provide both software and services, and sometimes hardware specifically designed for certain market applications. The GIS software market is approximately $250 million in size and is growing 30% per year. The software sales are causing significant additional expenditures on both hardware and services.

GIS can tackle problems that defy the capabilities of more conventional database management methods.

The GIS software market first appeared as a mainframe-based phenomenon. Those cartographers who needed to make maps usually worked for large government agencies in the civilian and defense world. The large, tailor-made mainframe GIS met their needs since it was designed to serve a largely "custodial" requirement.

TYDAC developed SPANS for end users, that is, professionals who wanted to use geographic information to help solve their problems. These professionals were computer literate, but not necessarily systems specialists with a custodial orientation. TYDAC first developed SPANS on DOS; now it has migrated to OS/2 and AIX. Unlike many other companies, TYDAC's product has grown up the computer environment ladder, rather than migrated down to the PC environment. Because of this, a certain discipline and emphasis on efficiency has characterized SPANS. Its analytical and modeling capabilities equal or exceed those found on the most complex mainframes or workstations.

In simple terms, SPANS can perform five major functions:

1. **Build and integrate spatial databases**. SPANS can accept and generate geographic information stored in three generically different data structures: vectors (lines), rasters (cells), and quadtrees (hierarchical cells). A digitizer or scanner that transforms paper map data into an electronic format can be used to input data into a GIS. Data files from other GISs and databases can be imported using a floppy diskette, magnetic tape, or optical disc.

This allows the analysis of maps, geographically referenced tabular data, and satellite images. SPANS has over twenty interfaces to existing GIS and database file formats, and links to graphics systems, spreadsheets, word processors, and statistical analysis packages.

2. **Explore relationships between spatial data sets.** Once a database is assembled, SPANS provides a battery of analytical tools to help users measure and understand the relationships between different geographic entities and theme layers. Models of reality can be constructed using this information and statistical methods.

3. **Query and identify suitable locations.** The ability to interactively query multiple layers of data in a spatial database by moving a cursor on a map is both an important user requirement and a strong feature of SPANS. With SPANS, it is possible to identify target areas in a geographic region that meet user-specified criteria.

4. **Assess impacts through models.** SPANS can define and visualize simple and complex models of dimensions of the real world that are strongly geographically dependent. These models can be defined within SPANS using its own modeling language or with user-defined subroutines.

5. **Build end user applications.** Under OS/2, SPANS can be configured and customized to meet particular user applications. A toolkit is provided to enable users to integrate GIS into their overall information management solutions.

SPANS has thirteen key modules assembled in an integrated menu as shown in Figure 2.

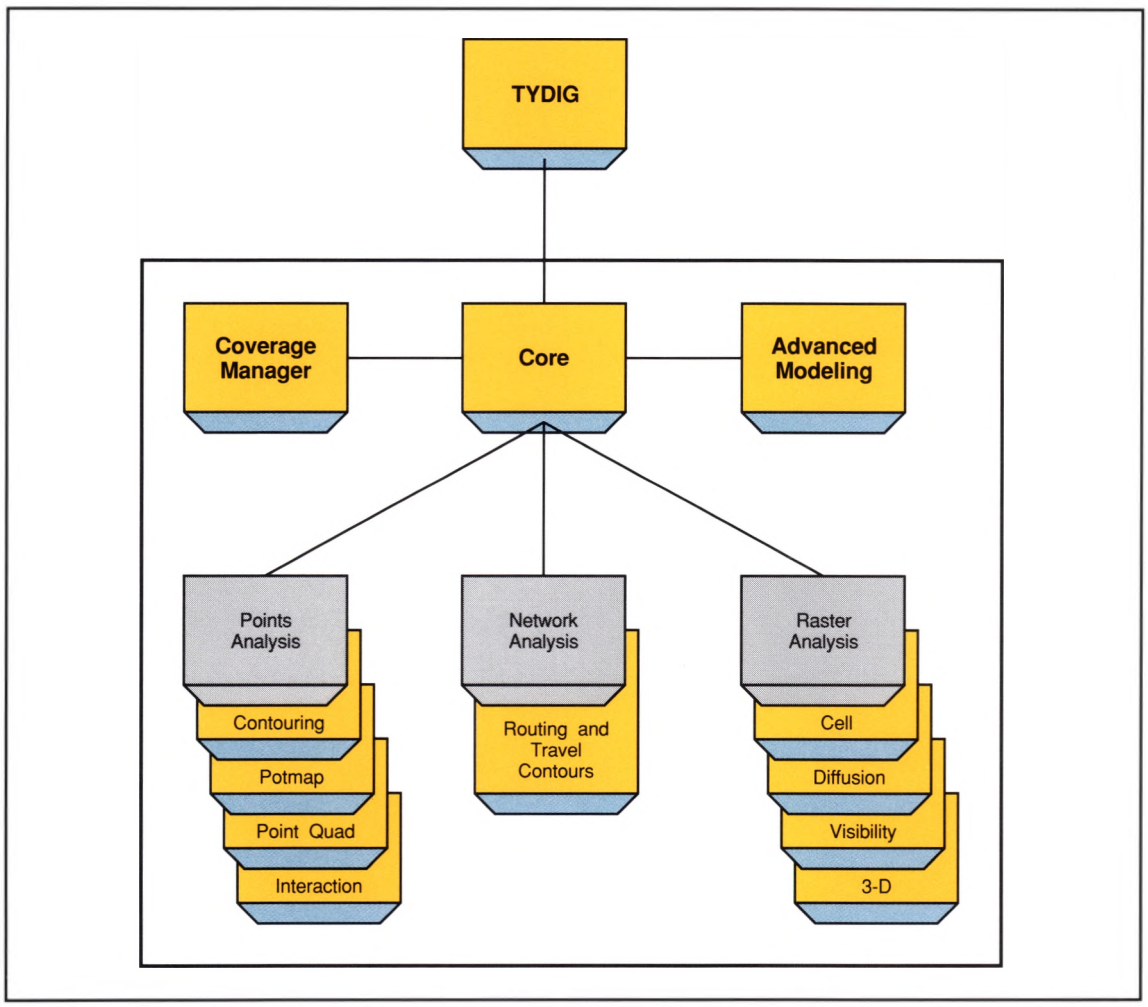


Figure 2. Overview of SPANS Modules

The modules can be packaged and assembled to meet the particular needs of different types of end users.

APPLICATIONS

The following examples demonstrate how SPANS can be applied to problem solving and decision making in many disciplines.

Retail

GIS can help businesses better understand their customers. For example, the market research department of a major national food retailing chain uses GIS to evaluate changing socioeconomic conditions surrounding their stores. This helps to compare store performance to the competition and to plan expansions and select new locations for future stores.

The retail chain used SPANS to construct a spatial representation of the key features (such as major roads, minor roads, parks, open spaces, location of subway stations) described in the market area where each store was located. Secondly, they incorporated into the GIS the socioeconomic data available from government and private sources (census tracts, trade areas, customer surveys) for the same area. Different sources of socioeconomic and market research data use different reporting units (or zones), but the geographic information system easily transformed this information into a common referencing system. This allowed the market researchers to quickly and visually cross-reference this information.

Once the database was in place, market researchers began replicating the normal manual analyses that they conducted to assess each store's performance. This often consisted of counting the number of people living within a certain distance of the store and calculating their sales potential. As the market researchers became more confident, they began to realize that their manual procedures had been restricted by their capability to manage data.

With this new GIS technology, the researchers could perform more sophisticated analyses that revealed insight into the structure and potential of their market. New methods for predicting customer interaction with the target stores are being investigated to calibrate more accurate models of store performance (see Figure 3).

Local Government

Managing urban growth is an important part of responsible fiscal policy for local governments. A key ingredient in accomplishing this is to ensure that the infrastructure necessary for development is in place. Alternatively, the growth to be accommodated by the expansion of the infrastructure could generate sufficient revenues to support the investment, but extensive modeling would be required before a local government would feel comfortable acting on this assumption.

GIS can help businesses better understand their customers.

Even before GIS, local government departments have used manual, and in some cases, computerized methods for determining costs of upgrading the infrastructure to accommodate growth. Based on this information, policies can be formulated about development approvals. It may, for example, be possible to expand the water and sewer networks to handle the growth. The resulting traffic from the new development, however, may create congestion of such magnitude

Figure 3. Before and After Store Expansion

SPANS is the first GIS to migrate to OS/2.

that it requires a significant investment in the transportation infrastructure. All the infrastructure impacts and the associated costs should be measured before proceeding. The difficulty, however, has been that these techniques were often highly geographically aggregated.

Using GIS, engineers and transportation planners in Montgomery County, a high-growth Maryland county, integrated many different types of data that enabled them to consistently evaluate the proposed development approvals. GIS also provided an insight into the spatial revenue structure of the county (see Figure 4).

SPANS was coupled with the county's transportation model to explore the interdependence between demographic, land use, and transport zones, and to generate better estimates of both jobs and households for smaller county subzones. The detailed parcel file maintained by the county tax assessor was used for this. The parcel file was address-matched. Analysis of this data helped researchers produce a better estimate of the percentage of households and jobs that were within walking distance of public transportation stops. This, in turn, increased the ability of the transportation model to simulate traffic patterns more accurately. SPANS also made it possible to help evaluate the long-range balance between the development permitted under current zoning and projected infrastructure.

The Federal Highway Administration is using this case study to demonstrate how state and local transportation planning agencies can use GIS for a wide range of issues including traffic, bridge and pavement management, hazardous material routing, and transportation demand modeling.

Environment

The National Oceanic and Atmospheric Administration (NOAA) is a major federal agency responsible for providing comprehensive information to help assess the effects of human activities on the nation's coastal oceans. The agency identifies and assembles data required for national and regional assessments of estuaries, ocean resources, and environmental quality. They communicate the resulting information to decision makers and institutions who must deal with multiple resource issues in the public or private sector.

Estuaries are important economically because they provide the nation with both goods and services. They are a transition zone between fresh water and marine ecosystems. Therefore, they play a recognized role in sustaining the health and abundance of the complex marine and animal life that resides in them. The challenge faced by NOAA was to construct an integrated spatial database that could assess the impact of human activities on the nation's coastal oceans.

SPANS was used to integrate data from a variety of sources. The first data set was sampled from over 5,000 maps compiled by another federal agency that illustrated wetland habitats according to a nationally established classification system (see Figure 5). The second data source came from NOAA itself, which included the best available information on the spatial and temporal distribution and life histories of species considered ecologically or economically important. The species considered in this inventory include fish, invertebrates, sea birds, water fowl, marine mammals, and turtles. The third data set required the integration of information regarding physical and hydrologic characteristics, habitats, land use, recreation, monitoring data on pollution sources, quality of shellfish-growing waters, hazardous waste sites, demographics, and pesticide use found in coastal wetland areas.

With this integrated database, government researchers used GIS to explore the relationships between particular species and ecosystems. Quantitative geographic models were developed to represent formally the relationships within these complex ecosystems. The models can now be used to assess the implications of possible changes in the quality of the physical environment on different habitats and species.

CONSIDERATIONS IN MOVING TO OS/2

SPANS is the first GIS to migrate to OS/2. Software developers do not switch operating systems without careful consideration. Why did TYDAC take this step?

Ease of Use

The reasons for TYDAC's move to OS/2 can be understood by first examining the company's marketing and technical objectives. TYDAC is dedicated to bringing GIS to end users. This means putting much effort into developing an increasingly easier to use and more approachable GIS that could be made accessible to a wide range of users. While TYDAC achieved a friendly menu-driven interface for the DOS version (relative to many other GIS systems that rely on command-driven operations), TYDAC saw that considerable improvement to the interface could and should be made. OS/2 1.2 offered a mature and rich environment for building and growing the product. Not only could SPANS' own interface issues be dealt with elegantly and simply, but through Common User Access (CUA), the resulting product would be compatible with the "look and feel" of other software ranging from spreadsheets and desktop publishing to database analysis and management systems.

Access to Processor Power

TYDAC was also concerned about the inability of DOS to handle the expanded functionality of the SPANS product. The memory limitations of DOS acted as a choke on the increasing power of PC platforms. OS/2 expands addressable memory to levels comparable to those available on workstations and allows the user to harness the inherent power of 80386 and 80486 processors. Memory management, which was becoming a serious development issue under DOS, disappeared with the adoption of OS/2. The prospect of OS/2 2.0 also offers a full 32-bit environment and continued growth potential.

Compatible with SAA

In the past, TYDAC has built its installed base by providing single seat DOS solutions to many customers. In positioning the company to offer GIS to the corporate community, it became clear that these customers needed the capabilities of OS/2 to efficiently disseminate multiple SPANS applications throughout their organizations. By operating under OS/2, SPANS can reach major clients that are committed to Systems Application Architecture (SAA). This compatible offering exploits the ever-growing list of communication and interface advances occurring between OS/2 systems and the large custodial database environment.

Access to SQL Capabilities

The availability of OS/2 Extended Edition (EE) Database Manager provided access to Query Manager, an excellent DBMS with Structured Query Language (SQL) capabilities. Together, SPANS and OS/2 EE supply a more powerful and consistent GIS platform.

Multiprocessing Potential

Looking toward the future of computer processor technology, it also was apparent that multiprocessing will become the cost-effective way to expand the power of personal workstations. TYDAC saw a strong foundation in OS/2 to exploit multiprocessing applications, initially through threading, and eventually through the Dynamic Data Exchange (DDE).

SPANS supports IBM's powerful new Intel i860-based Wizard card as an accelerator and coprocessor for computation-intensive applications. Initial tests of Wizard show that many SPANS operations will be ten times faster than standard PS/2 Models 70 and 80. Some SPANS functions that use the Wizard adapter include 3-D surface display and analysis, diffusion modeling, and network analysis.

SPANS supports IBM's powerful new Intel i860-based Wizard card as an accelerator.

Access to New Technologies

New technologies also are emerging on both the hardware and software fronts. These are important to TYDAC in its long-range development strategy. The ability to use some of IBM's new multimedia technology with OS/2 also provided TYDAC exciting growth potential for innovative GIS applications.

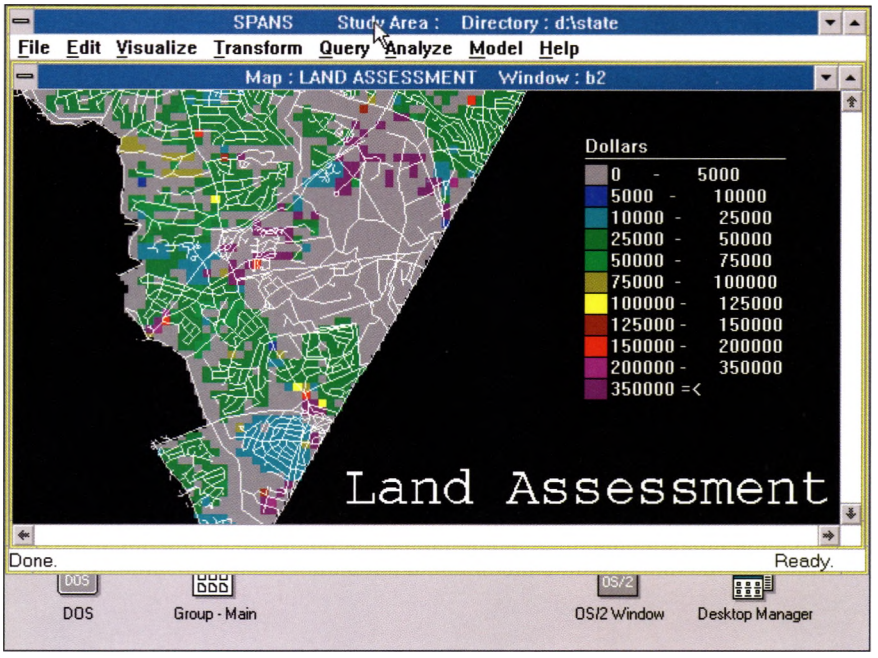


Figure 4. Land Assessment for Montgomery County

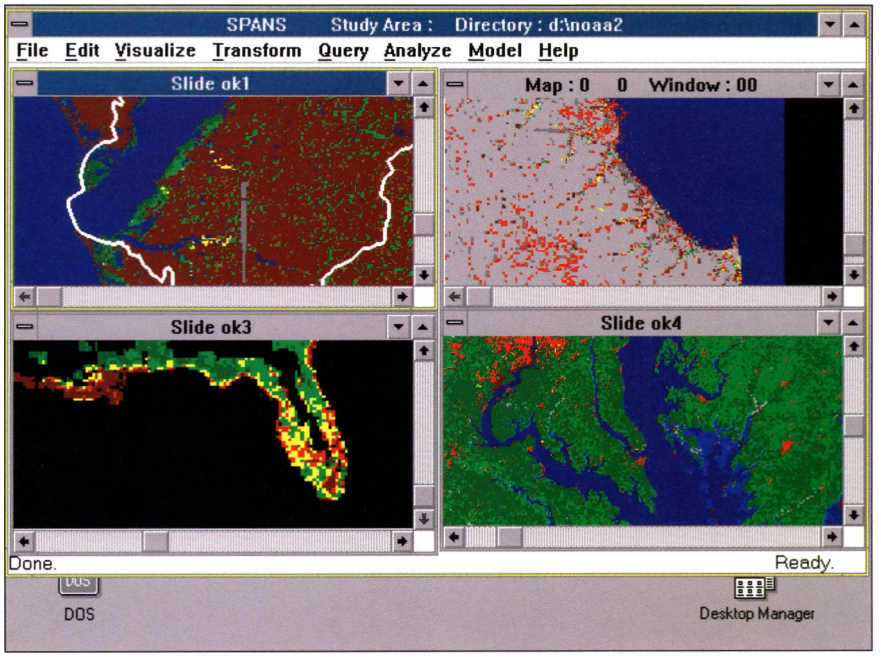


Figure 5. Wetland Habitats Classifications

Ideal Migration Strategy for Users

Finally, TYDAC determined that for its current DOS users, OS/2 was the most cost- and effort-effective solution in an upward migration strategy. By moving to OS/2, end users can achieve about 90% of the advantages of UNIX for about 10% of the effort over DOS. For example, the user learning curve going from DOS to OS/2 is naturally easier since OS/2 shares much of the same command set as DOS. With OS/2 2.0, users also gain access to the full advantages of the 32-bit world.

OS/2 is the upper end of the PC-based operating systems. It acts as an umbrella operating system that can accommodate DOS and Windows applications. Recent announcements indicating migration of Windows applications to OS/2 will expand the array of software available for OS/2.

APPLICATION DEVELOPMENT CAPABILITIES

A significant advantage that TYDAC offers in SPANS version 5 under OS/2 is the ability for users to construct their own applications solutions using SPANS as an engine. Using the facilities in SPANS and those available under OS/2, an application developer can combine a set of SPANS operations into one procedure or macro. This procedure can be activated under its own dialog box that offers the user access to those parameters necessary for executing the application. The application developer can activate SPANS models (in the SPANS modeling language) and, if necessary, link in other external C or FORTRAN-based subroutines.

OS/2's Dynamic Link Library (DLL) capability allows the SPANS user interface to be stored separately from the application. This allows for a multilingual product.

The user has control over the definition of the input parameters and the design of the dialog boxes, menus, and help prompts. Therefore, applications can be constructed for use by different language groups or in technical fields

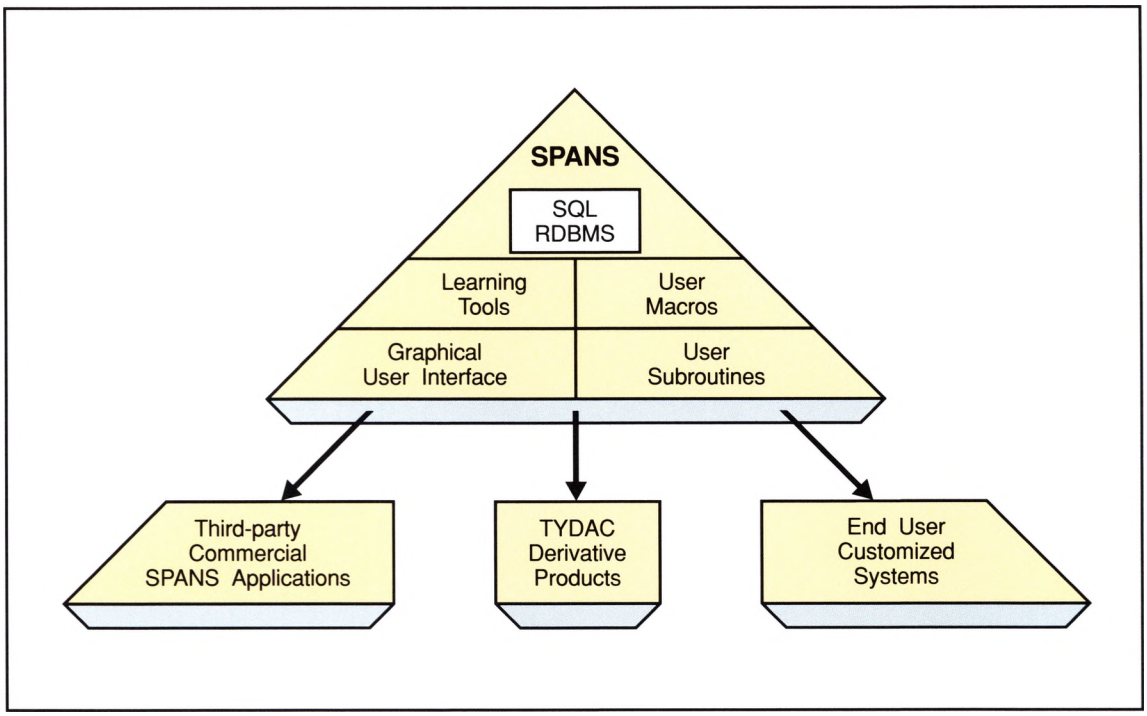


Figure 6. Application Development Concept

SPANS version 5 under OS/2 allows users to construct their own applications solutions using SPANS as an engine.

where the jargon and nomenclature are familiar to the end user. TYDAC provides this new facility to help clients and third-party developers apply GIS technology more quickly and efficiently to their particular fields (see Figure 6). TYDAC itself will be using this capability for building specialized products for target markets.

For example, a retail chain could select several stores and calculate the population income for areas closest to the individual stores. This can be achieved quickly by constructing a macro with three or four of SPANS' basic operations.

FUTURE DIRECTIONS AND OPPORTUNITIES

GSI is rapidly becoming an important element in the integrated corporate information environment. It is a technological development that has already proved its value. Through OS/2, it will become significantly more important to professionals and information managers in the business and resource management world.

Giulio Maffini, *TYDAC Technologies Corporation, 1655 North Fort Myer Drive, Suite 320, Arlington, VA 22209. (703) 522-0773. Mr. Maffini is president and co-founder of TYDAC. Before starting the company, he was executive vice president of the DPA Group, an economics and management consulting company. In this position he was involved with numerous development projects throughout Canada, Southeast Asia, the Middle East, and the Caribbean. Mr. Maffini graduated with honors from McGill University with a BSc and Bachelor of Architecture.*

Multimedia & Graphics

CD-ROM and Multimedia

Gary Barron

CD-ROM is the enabling technology for the multimedia applications of the 1990s.

by Gary Barron

In the multimedia environment, CD-ROM is an essential enabling technology. It addresses several issues that make distribution of and access to multimedia applications practical and economical. It is an attractive alternative for all PS/2 configurations, allowing a broad-based, low-cost platform for the distribution of software and information.

There is a growing popularity of interactive multimedia applications in networked or distributed environments. This requires an efficient and economical means for reproducing and distributing information to users in multiple locations. Among the new technologies available for this purpose is Compact Disc Read-Only Memory (CD-ROM), which can store over 550 MB of data.

CD-ROM hardware adapter cards and software drivers have been available on PC Bus and Micro Channel IBM platforms through OEM suppliers. Most users have used this technology primarily in the DOS environment. Recently, IBM announced its own internal and external drives and adapter cards for CD-ROM. Currently, these drives are supported only by the Micro Channel Small Computer Systems Interface (SCSI) adapter cards. PC Bus configurations, such as the Model 30, can currently be accommodated by OEM SCSI cards and OEM CD-ROM drives.

According to the IBM U.S. Marketing & Services's product announcement on March 20, 1990, the CD-ROM drive is recommended for customers who have or plan to order PS/2 Micro Channel desktop or floor-standing systems, and who need the function and capacity of a CD-ROM drive. No additional system slot is required because multiple external SCSI devices can be connected to a Micro Channel SCSI adapter in a "daisy chain" manner. All floor-standing models announced since March 20, 1990 have included the SCSI adapter built into the system board. The PS/2 CD-ROM drive is for both large and small customers who require applications or data distributed on CD-ROM media, such as catalogs, indexes, and reference data (dictionaries, encyclopedias, or other data).

With the growing demand and integration of quality images—audio and video into desktop applications, reference materials, and training applications—the CD-ROM is the enabling technology for the multimedia applications of the 1990s. As we design and develop applications with enhanced user interfaces using multimedia, the CD-ROM technology is a key component of the delivery system. To evaluate more effectively what CD-ROM can bring to the multimedia application environment, we should look at this technology from the viewpoints of the user, the publisher, and the application designer.

THE USER'S VIEWPOINT

Consider an environment where users need to access a multitude of courseware titles or multimedia applications. They must download from 5 MB to potentially over 200 MB of data, images, and audio files. This is not practical for diskettes or host-connected systems. (At 1200 baud, it would take forty-six days, non-stop, to transmit the contents of one CD-ROM). The IBM 3330 Optical Disc Drive, using the Write Once Read Many (WORM) technology, has a capacity of 200 MB of storage and can be used to distribute applications to users. The cost of the hardware and the media alone is significantly greater than CD-ROM technology. The biggest factor is the cost of writing or copying an application to the WORM cartridge. This must be done with a computer; it cannot be stamped like a

CD-ROM. As a result, it is not practical to publish and distribute single titles on this media.

The single CD load-and-go approach for a user is the simplest approach to executing individual applications on demand at a workstation. WORM technology, therefore, does not provide the user with a cost-effective load-and-go environment. An application on a CD-ROM can be integrated onto a workstation as another application, with the major part of its files, graphics, and audio accessed directly from the CD without a lengthy download to the hard disk before execution. With cases such as reference material, there may be no download required.

With the announcement of the IBM CD-ROM drives (both external and internal to the PS/2), IBM and Microsoft jointly developed a CD-ROM device driver for OS/2. As a result, DOS applications can be executed within the OS/2 DOS box, or OS/2 applications can be run without any special language or operating system commands.

There is one dominant exception that deals with playing a mixed-mode (that is, ISO 9660 data and Red Book CD-audio recording on the same CD) CD-ROM. This play feature (the same as playing home audio) has been integrated into several titles available in the market on DOS. It is currently not possible to execute the CD-audio portion of the program in the DOS box of OS/2 because of some direct device initiation through the DOS BIOS. The DOS box driver support is under OS/2 system constraints and does not allow direct manipulation of a device. This issue needs to be addressed for compatibility of DOS applications using CD-audio to run under OS/2 in the DOS box.

A single CD-ROM can be dedicated to each individual title with extensive room for growth, or to multiple course segments with multiple language versions. The multiple languages issue is best addressed through using digital audio files that can play up to thirty hours of AM quality audio using the IBM Audio Capture and Playback Adapter (ACPA) at voice level on a single CD.

Generally, less than 5% of the application needs to be downloaded to the hard disk for student control, course tracking, and any data updates of the runtime software. This means the initial load is very short—generally fifteen to thirty seconds. Subsequent executions can be started without downloading. Hundreds of titles can be executed on the smallest PS/2s with minimal impact on the hard disk.

THE PUBLISHER'S VIEWPOINT

CD-ROM provides a durable and highly reliable media compared to diskettes or communication networks. Documentation, application programs, and databases can be distributed on the same media.

Labels with the release numbers and dates can be printed on the CD-ROM surface. One advantage of a read-only medium is that users cannot modify the program. When many users must be updated, restamping with updated labels provides a precise, externally verifiable label. The size of the CD-ROM is small enough for minimal mailing expense. Installation usually consists of replacing the current copy in the user library, or a minimal download procedure.

Low-cost Alternative

The external IBM CD-ROM drive costs under $1,550 (less if volume discounts are available). The cost of replicating CD-ROMs is about $1,500 for mastering and about $3.00 per disc in reasonable volumes. The adapter card used to connect the CD-ROM drive can be a SCSI card. The SCSI interface is a parallel adapter port similar to the RS232 interface port, but much faster. IBM offers a 16-bit SCSI adapter card and a 32-bit SCSI adapter with a 512 KB cache. A SCSI port can support up to seven SCSI devices. The PS/2 Models 65, 80-A21, and 80-A31 have the SCSI interface built into the motherboard.

The following characteristics make CD-ROM a low-cost alternative:

Read-only Medium: Because the data on a CD-ROM cannot be updated or copied easily to another CD-ROM, it is ideal for the publishing industry. This allows a publisher to license and distribute titles for a fee without concern about pirated versions made by users. This does not mean that individual programs, data, images, or audio could not be copied; but the entire program could not be easily replicated and reproduced.

IBM and Microsoft jointly developed a CD-ROM device driver for OS/2.

The Compact Disc File Support (CDFS) system has been developed for OS/2.

Low Cost: Since the disks are stamped or pressed rather than written magnetically, the cost of making copies is low. Minimal time is required to produce high volumes of discs of the same title. For this reason, it is practical to update 1,000 to 10,000 copies and mail within a short time period, with approximately a 48-hour turnaround for pressing time.

Reproducing WORM technology (tape or diskette) requires a computer-write operation. This is extremely costly and time-consuming when replicating 50 MB to 200 MB applications. This size of application is common for just-in-time education applications using digital audio, high-resolution graphics, and digital video.

Durability: Because of the tough polycarbonate base and lacquer coating, data damage almost never occurs through normal handling of the media. Estimates for CD-ROM media life exceed half a century.

An International Standard

CD-ROM media is defined by an international standard for hardware audio and data. This facilitates international application development. In addition to these standards, Microsoft has developed an extension specification that allows a user to access data in a standard DOS environment, similar to any other read-only data on hard disk or diskette.

A similar file support system, referred to as the Compact Disc File Support (CDFS) system, has been developed for OS/2. This file support system driver is delivered on the IBM OS/2 option diskette with the CDROM.SYS driver and the SCSI.SYS driver. Microsoft has also announced this CD-ROM file support system and its specification for developers. This means it is not necessary to modify software that accesses data in a normal DOS environment. The logical path in DOS or OS/2 should be modified to the alternate logical device (drive L: rather than drive C:).

The device driver for DOS, known as MSCDEX.EXE, is called the Microsoft DOS extension for CD-ROM. Adapter card developers, as well as OEM CD-ROM drive manufacturers, have developed their hardware drivers to interface properly to the Microsoft DOS extension specifications for several years now. This is another major adaptation that allows developers to write code in the DOS environment without special coding considerations for the CD-ROM media. The application designer should still allow for read-only or performance considerations that a hard disk design option may provide.

The DOS extension MSCDEX.EXE, while available from Microsoft, is normally packaged with the CD-ROM hardware driver software. The cost of this driver is generally embedded into the cost of the hardware and associated hardware drivers. This also is true for the IBM CD-ROM driver software.

For the OS/2 operating system, IBM and Microsoft have jointly developed the CD-ROM file support system driver. This file support driver is delivered on the option diskette with any CD-ROM driver order. Both the OS/2 and DOS hardware drivers are on the option diskette along with the installation procedures.

THE DESIGNER'S VIEWPOINT

As graphic imaging has improved over the past few years from CGA to EGA to VGA, the size of images has increased dramatically. The typical screen image buffer for a bit-mapped image has grown from 16 KB to 64 KB to 720 KB, and will be even greater with Extended Graphics Array (XGA). Even with image compression techniques, storing these images on disk is expensive. CD-ROM allows developers to provide large quantities of high-resolution images cost-effectively on any PS/2 model.

An example of this is the "Media Source General Topics Library," volume 1 developed by Applied Optical Media Corporation in Malvern, Pennsylvania. This CD contains 1,500 full-screen VGA images (640x480x256 colors) with approximately 500 segments of audio sound effects and background music. The software interface to select and download image and audio objects is also on the CD. This is possible only on the CD media for cost-effective distribution.

Access to Large Databases

Large databases of over 50 MB can be accessed easily and directly from the CD-ROM, even on PS/2 Model 25s. CD-ROM can eliminate the need to be online to a host database for access to static data. Most multimedia data-

bases (that is, bitmapped images, digital audio, descriptive text) are static. Our experience in converting seven InfoWindow multimedia applications to CD-ROM showed that less than 5% of the data was dynamic. This data could easily be maintained on the hard disk. Dynamic databases on a PS/2 or host can be integrated into the application, if required, through the normal logical assignment of files to these devices. In some circumstances, a file selection through the operating system path function can be used to override CD data on an exception basis.

Placing large, static databases on CD-ROMs can sometimes avoid line costs and also provide local access when a host is not available. Many databases, especially in just-in-time education and reference material (Microsoft's Bookshelf, Census Bureau demographics, and so on), seldom need updating.

Graphics can be displayed on the screen much faster from a local device than by transmission over communications lines from a host system. This can be critical for establishing adequate application performance in an interactive application.

Ability to Grow Applications

Data storage requirements typically grow over time for applications such as employee benefits or inventory reference. This growth can include increased amounts of data plus images and audio. CD-ROM allows a designer to start with a small application and expand it over time, yet still maintain a single CD-ROM configuration. Many applications can be designed and contained within the 550 MB capacity of the CD-ROM. This is often true for reference material, generic databases, and interactive training applications.

With up to seven CD-ROM drives attached to a PS/2, a large amount of material is available to applications. For some extremely large database applications, a custom design can easily address up to seven CD-ROM drives through logical assignments on a single SCSI adapter port. Although it would not be practical for a publisher to distribute multiple online CD applications, the hardware and operating system are easily adapted for unique design requirements.

ACCESS TO CD AUDIO

The audio capability of the CD-ROM allows the designer to overcome the audio limitations of the laser disc in an interactive video application. Laser discs provide two thirty-minute audio tracks that provide an analog audio output. This amount of audio is often not adequate for an application that can include several hours of audio playing over still images on the laser disc, or graphic images on animation.

Since CD-ROM disc format is compatible with CD audio, audio CDs can be played on a PS/2. If the CD drive has an audio output, using this audio source does not add any incremental hardware cost to the system. That is because it is played directly from the CD audio output into the same amplifier source used by the system. Most vendors today provide an audio-output option on CD-ROM drives, including the announced IBM drives.

CD-ROM allows the designer to overcome the audio limitations of the laser disc.

Today, it is possible to stamp a mixed-mode CD that produces standard CD audio in one area of the disc and digital data sets for normal DOS and OS/2 program accessibility in another. Although IBM provides an audio play utility for DOS and OS/2, this access has not been implemented in any of the high-level authoring tools within IBM. Some CD titles currently use this technique very effectively. The result is 72 minutes of stereo or 144 minutes of high-quality CD audio control without the need for an audio card on the system board.

An example of this is the "Multimedia Birds of America" CD produced by OMC Research Inc. of Portland, Oregon. Another example is the recent announcement of the "National Geographic Encyclopedia of Mammals" that was produced using the LinkWay authoring system. Keep in mind that both of these applications will not play the audio portion in the DOS box of OS/2 because of the limited audio driver support previously discussed in this article.

CD-ROM/XA

The International Standards Organization announced at the March 1990 Microsoft CD-ROM Conference in San Francisco that the CD-ROM/XA audio specifications are now a firm standard, although the graphics

CD-ROM can support a technique for displaying limited motion video on a PS/2.

standards have not yet been settled. This is significant because there is now a worldwide standard specification for uncompressed digital audio at four different levels of recording on a CD-ROM. This includes the current CD audio standard of 72 minutes of stereo, which has allowed worldwide development of players that can play CD audio music. The three additional levels of audio allow you to record over 19 hours of audio at various quality levels.

Providing a smarter chip for the digital-to-analog conversion of audio through a digital signal processor enables all four levels to be used, based on quality and volume requirements. By adapting to this standard, application developers could digitize their audio files for use on multiple platforms for worldwide distribution.

At this time, IBM has not adapted its hardware or software to this standard. Current implementations by other vendors have only been through a black box converter outside the SCSI adapter or another board. As this implementation of audio becomes accepted and designed into the system, it can provide a low-cost implementation of a wide range of audio frequencies on a personal computer.

CD-ROM also can be used as a storage source for digital audio as provided by the IBM Audio Visual Connection (AVC) ACPA card. Prerecorded digital audio data sets can be stored on a single CD for playback during an application at three different audio levels:

- Voice Level: 5.5 KB/sec (thirty hours of audio)
- Music Level: 11 KB/sec (fifteen hours of audio)
- Stereo Level: 22 KB/sec (eight hours of audio)

The IBM M-Motion Video Adapter card also provides a digital audio capture and playback feature at 7.85 KB/sec. This allows over twenty hours of audio to be stored on a single CD. CD-ROM allows for a broad design range of data capacity or a large capacity for future growth.

Motion/Animation Option

CD-ROM can support a technique for displaying limited motion video on a PS/2 by refreshing compressed frame images on the screen at ten to twenty frames per second. This can be effective when shown in partial screen images to increase quality and extend video time. For instance, a 64 KB Monochrome Color Graphics Adapter (MCGA) full-screen refresh from a buffer on a 16 MHz PS/2 cannot produce more than two to three frames per second. The effect would not be adequate for animation. On the other hand, a developer could probably refresh one eighth of a screen in this mode at perhaps ten or more frames per second, and obtain a low-quality video-like sequence. Conceptually, Digital Video Interactive (DVI) uses a high-frame compression algorithm with a frame delta to establish high-quality video feed at a slow transfer rate (150 KB/second) from the CD-ROM.

There are experiments in the marketplace with several adaptations using various compression techniques, image sizes and resolutions. Sound also can be provided at the same time. This technique was incorporated into the "National Geographic Encyclopedia of Mammals" CD. Techniques similar to CD-ROM/XA are being tested to interleave audio with image data to provide sound with a slow frame rate for movies with sound using partial screens.

Another technique for audio with animation or motion video is being developed through the CD-ROM/XA specification. This new format specification allows interleaved data and audio so that a single access of the laser can play sound over animation by picking up a sound sector alternately with data sectors. This can provide 640x480 256-color video with sound for windows less than two inches. This is done without compression. This has not yet become a standard feature on a Micro Channel or the PC Bus adapter card.

Gary Barron, *IBM Multimedia and Education Division (M&ED), 4111 Northside Parkway, Atlanta, GA 30327. Mr. Barron has been with Multimedia Marketing Support for the past five years. Currently a senior marketing requirements analyst, he has been with IBM since 1969 with assignments as a systems engineer, product planner, and systems engineering manager before working with Multimedia Marketing. Mr. Barron holds a BS in Business Administration from California State University at Fresno.*

Multimedia & Graphics

An OS/2 Business Graphics Toolkit

by Jim Crossley

The Developer's Business Graphics Toolkit from The Crossley Group, Inc. is a toolkit to help Presentation Manager (PM) application developers integrate two- and three-dimensional business graphics into their application programs. The heart of the toolkit is an intelligent graphics engine provided as a Dynamic Link Library (DLL). High-level Application Programming Interfaces (APIs) are issued to bridge to this DLL. Business graphics can be incorporated into new or existing applications with minimal code supplied by the developer. The high-level APIs eliminate the need to learn over 200 PM Gpi function calls. This new product was designed to ensure end user appeal, ease of developer implementation, and exploitation of the OS/2 hardware through PM. The Developer's Business Graphics Toolkit was built specifically for OS/2. It is not a migration from another platform.

The Developer's Business Graphics Toolkit version 1.0 allows the developer to select from many chart types, shown in Figures 1 and 2. The scatter, line, smooth line, and area charts are presented in a two-dimensional display. The bar, bar column, stacked bar, and pie charts can be presented in a two- or three-dimensional display. The perspective bar chart is a three-dimensional presentation.

The Y-axis and Z-axis support manual or automatic scaling for positive and negative data, including floating point. Automatic X-axis application-specific element support includes the following:

- General numbering (0 to n)
- Calendar (days, weeks, months, years)
- Time (seconds, minutes, hours)
- Military/International hours
- Accounting cycles (30, 60, 90, and so on).

Data is summarized automatically and percentages displayed for pie and column charts. The developer can specify chart text, such as titles, footnotes, margins, and data legends. Fonts resize automatically in the client chart areas. Besides using the standard sixteen colors, OS/2 developers can specify any RGB color, of which their display is capable, to enhance the graphic chart.

The Developer's Business Graphics Toolkit is a DLL module that minimizes the application program size. The toolkit supports many languages and tools currently in use with OS/2, including C, COBOL, the Dialog Manager, as well as IBM's National Language Support.

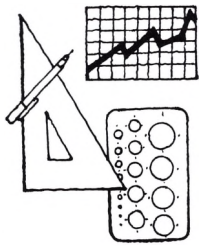
Besides a well documented and robust programming interface, the Developer's Business Graphics Toolkit provides the application developer with over forty sample programs (including source code and chart icons) for education or rapid prototyping efforts. The demonstration is an interactive, hands-on program that allows the OS/2 developer to change many parameters, as shown in Figure 3. End users will find this rapid prototyping capability useful.

CHART BASICS

The developer specifies whether graphic charts are presented in a window or portion of a window. Charts can be shown in a com-

Jim Crossley

The toolkit supports many languages and tools currently in use with OS/2.



plete window with other application data, in scrollable windows, or in dialog boxes. Creating a chart instance is similar to creating a standard PM window using the ChartCreate API, as shown in Figure 4. The developer provides this API with a handle for the PM window and the coordinates within the window to display the chart. Defaults are provided for the percentage of space allocated to headings, footings, margins, and the central graphic area. Developers, using their own percentages, can dynamically change the amount of space assigned to these areas to meet their specific goals. The DLL automatically handles all scaling of the text and graphic images.

Using data arrays, the developer then provides data to the chart. The chart supports a maximum of eight data arrays (representing groups) and 2,048 values per array (representing points). Data arrays supported include SHORT, USHORT, LONG, ULONG, and float data types. Each data group requires a separate request through the ChartData API. The graphics engine retains the specified data in dynamic storage until the data is changed or the chart is destroyed. Besides specifying a pointer to the data array, the developer can provide an ASCIIZ string that represents the text legend for each data group.

The ChartSetXTicks API provides automatic X-axis text values. For example, if the data to be presented reflects the last three months of 1989, the developer can specify that the X-axis be displayed in months, starting with the tenth month. The chart shows the months of October, November, and December. If the developer provides more than three data points, the chart continues with January as the fourth month and follows through the next year as long as data is provided. If the chart does not have enough width to accommodate the entire spelling of the months, the DLL shifts into an abbreviation mode, such as "Dec" or "D" for the month of December. When the DLL does not provide the type of text required, the developer can provide a text array that can serve as an extension to the graphics engine.

The developer can provide the chart heading, footing, left margin, and right margin text to the chart using one request to the ChartSetTexts API. These also can be specified individually through a ChartSetText request. The developer issues a ChartVisible API call to make the chart visible to the user.

There is a ChartSetRect function call to help the developer support the user's ability to resize a window. This call informs the graphic chart that the amount of real estate for the chart has changed. The chart is then redisplayed through the ChartVisible or ChartSetType function request. The ChartSetType API enables the developer to change the current type of chart to another chart type. Since the data, headings, and other information have been retained by the DLL, no additional steps are required.

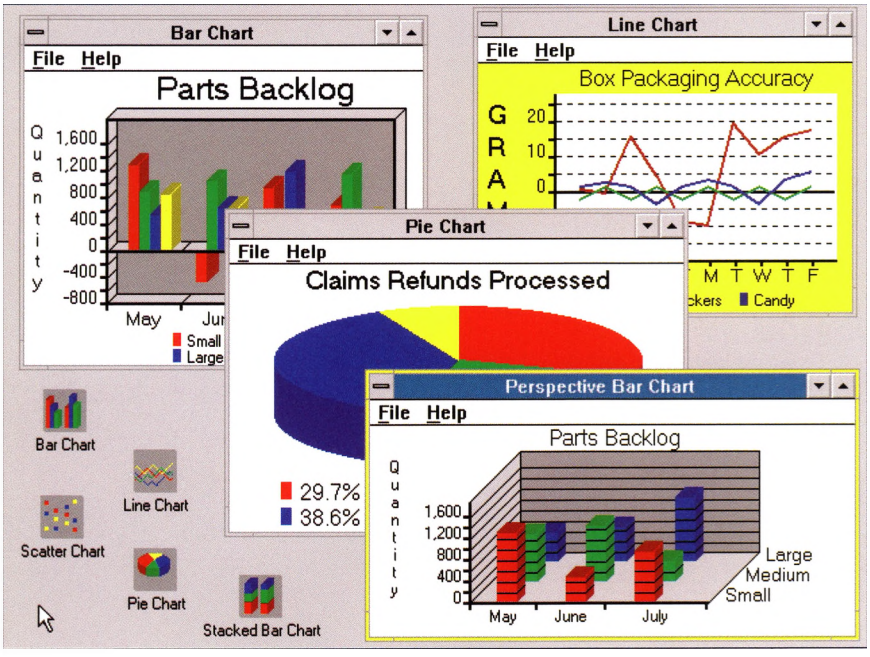


Figure 1. Sample Graphic Representations

Coloring the Chart

Each data group has a specific pair of colors assigned to it by default. For example, data group one is assigned red and dark red (for the shadowing effects of 3-D charts). The developer can change these defaults to any of the standard sixteen colors or specify RGB values to be associated with a data group.

Colors can also be specified for all other portions of the chart. This includes, but is not limited to, the text, tick marks, chart background, and the graphic background.

Calculations

Besides extensive automatic scaling facilities, the DLL will provide automatic summations of data to display percentages for the column and pie charts. The developer does not have to perform these summations or calculations. For example, to change a bar chart to a pie chart, the developer simply calls the ChartSetType function call. The data provided to the bar chart is recalculated, summarized, and then displayed as a pie chart. In support of the stacked bar and column charts, the graphics engine will sort the data by ascending magnitude if requested by the developer. This sort, performed in memory, requires no disk files or disk activity.

Text and Legend Placement

The graphics engine centers and aligns all text within the chart area. It does this through internal font size balancing algorithms. It is dependent on the amount of graphic space available. For example, a chart consisting of four data groups or legends may show the four legends placed side by side, or in two groups of two. The placement is based on such considerations as available chart width, height, and the characters in the legend text. The DLL will sacrifice certain text when the window is too small for the smallest system font. The DLL's goal is to provide as much data as possible in the space available.

PM Messages and When to Paint

The graphics engine, or DLL, does not post or send any PM messages to the developer's application. The developer must specify when to repaint the graphic chart through the ChartVisible or ChartSetType APIs (see Figure 5). The graphics engine does not create any PM windows to accomplish its tasks. It uses only Gpi function calls to achieve its goals. The ChartDestroy API removes the chart instance, as shown in Figure 6. The application window belongs to the developer and the graphics engine is merely a high-level support service acting as an extension to PM.

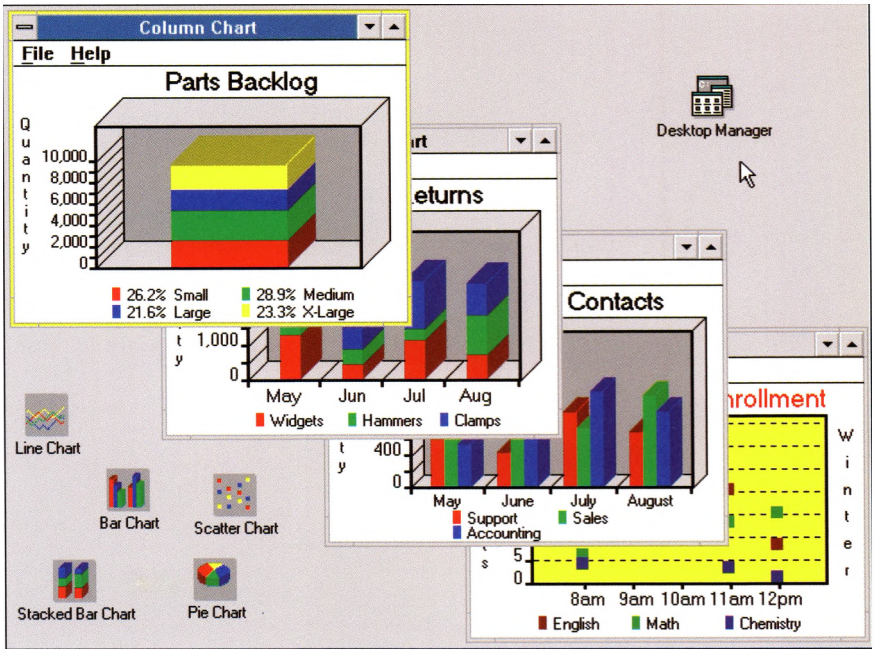


Figure 2. More Graphic Representations

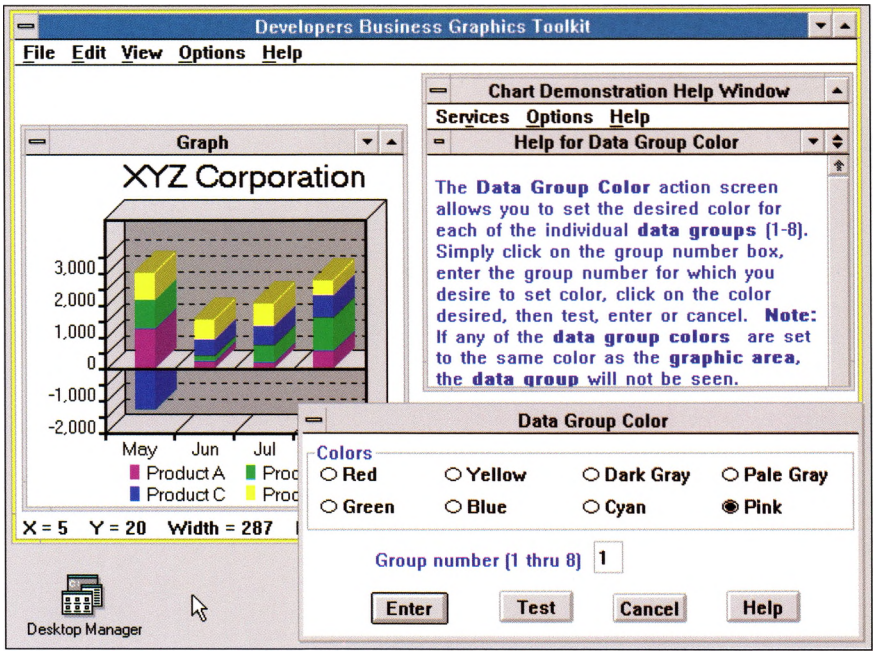

Developers Business Graphics Toolkit
File Edit View Options Help
Chart Demonstration Help Window
Services Options Help
Help for Data Group Color
Graph
XYZ Corporation
3,000
2,000
1,000
0
-1,000
-2,000
May Jun Jul
Product A
Product C
X = 5 Y = 20 Width = 287
The Data Group Color action screen allows you to set the desired color for each of the individual data groups (1-8). Simply click on the group number box, enter the group number for which you desire to set color, click on the color desired, then test, enter or cancel. Note: If any of the data group colors are set to the same color as the graphic area, the data group will not be seen.
Data Group Color
Colors
Red
Yellow
Dark Gray
Pale Gray
Green
Blue
Cyan
Pink
Group number (1 thru 8) 1
Enter
Test
Cancel
Help
Desktop Manager

Figure 3. Demonstration and Rapid Prototyping Tool

The graphics engine centers and aligns all text within the chart area.

```
/*        Chart data, pointers to data and group legends                */

static SHORT     ausData1[] =  { 1237, -425,  900,  650 } ;
static SHORT     ausData2[] =  { 850,  1000, -350, 1100 } ;
static SHORT     ausData3[] =  { 500,   600, 1150, -300 } ;
static SHORT     ausData4[] =  { 800,   600,  250,  450 } ;
static PSHORT    apDatap[4] = { ausData1, ausData2, ausData3, ausData4 };
static PCH       aszLegends[4] =  { "Small", "Medium", "Large", "X-Large" };

   switch (msg) {
      case WM_CREATE :

       /* Chart Initialization is performed here.                        */
       WinQueryWindowRect(hwnd, &ClientRctl); // Determine window size will
                                              // return 0's

       /* Create the Bar Chart Object                                    */
       hChart = ChartCreate(hwnd,    // Client window
          CH_BAR,                    // Bar Chart
          CH_X_GRID,                 // Chart flags, (horizontal grid lines)
          &ClientRctl,               // Client rectangle area
          &usErrors,                 // Pointer to error location
          CH_DATA_SHORT,             // Type of data for chart
          4,                         // Number of data groups
          4,                         // Number of points
          3);                        // Number of dimensions

       /* Specify calendar months for x-axis starting with May           */
       ChartSetXTicks(hChart, CH_WIDTH, CH_MONTHS, 5);  // Start with May

       /* Set the chart title and left margin texts                      */
       ChartTexts(hChart,"Parts Backlog",NULL,"Quantity",NULL);

       /* Set the area behind the bars to dark gray to add more dimension*/
       ChartSetColor(hChart,CH_GRAPHIC,CLR_DARKGRAY);
       break;
```

Figure 4. Creating a Chart Instance

```
case WM_PAINT :

   /* Perform developers normal window programming                       */
   hps = WinBeginPaint(hwnd, NULL, NULL);  // Standard PM developer
   GpiErase(hps);                          //   code, validating
   WinEndPaint(hps);                       //   the window

   /* Obtain the new window size and repaint the window                  */
   WinQueryWindowRect(hwnd, &ClientRctl); // Determine window size
   if (ChartSetRect(hChart,&ClientRctl)) {

      /* Provide the Bar Chart with some data to chart                   */
      for (i = 0; i < 4; ++i) {  // Loop through the groups
            ChartData(hChart,    // Chart handle
            i+1,                 // Data group number
            apDatap[i],          // Location of data
            4,                   // Number of data items for group
            0,                   // X-axis starting point index
            CH_DATA_SHORT,       // Type of data to chart
            aszLegends[i]);      // Legend for the group
      }

      ChartVisible(hChart);      // Make the chart visible
   }

   break;
```

Figure 5. Displaying the Chart

```
case  WM_CLOSE :
case  WM_QUIT :

   ChartDestroy(hChart);          // Close out the chart
   break;
```

Figure 6. Destroying the Chart Instance

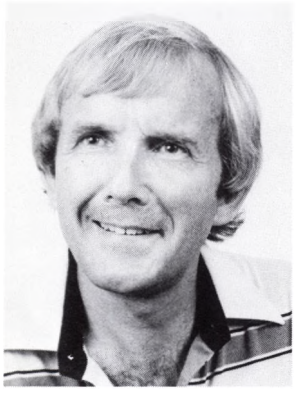
SUMMARY

The Developer's Business Graphics Toolkit version 1.0 provides the PM programmer with a quick and easy method of incorporating eye-catching and very pleasing business graphics into an OS/2 Presentation Manager application. This tool is for the developer who needs effective business graphics to be integrated with new or existing applications with as little effort as possible. This allows the developer to concentrate on the main ideas and objectives of the application without having to worry about manipulating and presenting the data.

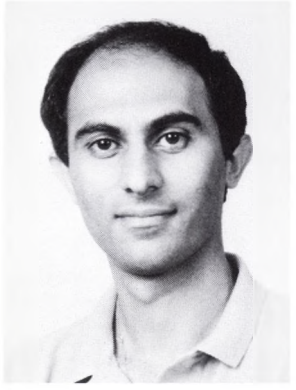
Jim Crossley, *The Crossley Group, Inc., P.O. Box 921759, Norcross, GA 30092. (404) 751-3703, Fax (404) 751-3704. Mr. Crossley is the president and head of research and development. Previously with IBM, he held positions of worldwide chairman of the Process Control User's Group, manager of sensor-based engineering and technology, and architect of network hardware. During his last eight years in Atlanta, he was a member of the Laboratory Director's advanced technology staff. His last assignment was as a Systems Application Architecture (SAA) marketing technical consultant with primary emphasis on OS/2 tools and technologies. He has received awards for Outstanding Contributions and Outstanding Innovations. Mr. Crossley holds a local-area-network software patent.*

Multimedia & Graphics

Corel Draw Migration to OS/2 PM



John McFetridge



Razmik Malek-Adamian

by John McFetridge and Razmik Malek-Adamian

Corel Draw is a full-function illustration package developed by Corel Systems. Originally targeted for graphic artists, it has proven to be a powerful tool for anyone wanting to enhance their document quality. The OS/2 version of Corel Draw evolved from the Microsoft Windows version, which is based on an earlier Corel product called HeadLine. This article describes some techniques used in porting the Windows version to OS/2.

The Windows version of Corel Draw, released in February 1989, took ten programmer-years to finish. It has been a resounding success, selling over 80,000 copies worldwide. The original English release has been complemented with national language versions in German, French, Spanish, Italian, Dutch, and Swedish. In the past year, Corel Draw has won several industry awards, including the *PC Magazine* Editor's Choice, *Personal Publishing* Drawing Product of the Year, *Publish!* Reader's Choice, and *PC World*'s Best Buy. Figure 1 shows an example using Corel Draw to create a color rendition of a diver from a wire frame.

In June 1989, we began to port Corel Draw to OS/2's Presentation Manager (PM). At that time OS/2 1.1 was already shipping. With OS/2 1.2 just around the corner, we decided to develop on the 1.2 platform. We moved code directly to PM. We skipped porting tools, such as Mirrors and the Software Migration Kit (SMK), because we felt Corel Draw's graphical routines demanded direct control for best performance. Also, by writing directly to OS/2, we could later take full advantage of OS/2 2.0's powerful features, such as its non-segmented architecture. Since we had little previous knowledge of Windows, PM, or Corel Draw itself, the learning curve was extensive. We spent much time cross-referencing Windows and PM manuals.

We first concentrated on developing tools to simplify moving the Windows code to PM. We tried to blend those two programming environments, while leaving the door open for future ones, such as MOTIF. We wanted to avoid the use of conditional compilation statements, such as IF DEFINED OS/2, by mapping many Windows Application Programming Interface (API) calls to PM with macros and subroutine calls. We wrote utilities to translate Windows resources, such as dialog boxes, into their PM equivalents. While this initial porting effort took about eighteen programmer-months, we feel that a future release would take only two to three months to move from Windows to PM.

COREL DRAW FOR OS/2

Although Corel Draw takes full advantage of the PM Graphical Programming Interface (GPI), we still wrote some of our own routines, such as PolyBezier calls and scalable font support. We developed our own Bezier technology to add functionality to the Windows graphic programming model; we carried that forward in the PM version for performance reasons. Because we used our own technology, the code can be moved later to other platforms that do not provide such support.

The OS/2 1.2-level scalable fonts were not deemed rich enough in quantity nor functionality for Corel Draw (see "Adobe Fonts Selected for IBM OS/2 and SAA" in the Summer 1990 *IBM Personal Systems Developer*). So we ported our own fonts created in the Windows

version to correspond to 102 industry-standard typefaces, such as Times Roman. Corel Draw uses these vector fonts on the display screen. If the target is a PostScript® printer, we use the fonts resident in the printer. If a font is not downloadable, or if the printer is not a PostScript printer, then we use the same vectors to create the printed characters.

Because we found some OS/2 PM graphics primitives that performed better than their Windows counterparts, we did not need new customized routines. For example, we were pleasantly surprised by the speed of the PM routine for defining Clipping Paths.

PM has proven to be an excellent development environment. Multitasking is a real boon to developers, especially those familiar with minicomputer and mainframe environments. It allows us to work on other things, such as editing, while a compilation is occurring simultaneously. An improved CodeView® made our lives much easier. Although CodeView had some problems debugging large Windows applications, the OS/2 version addressed most of these. We also made extensive use of the MultiScope® debugger from Logitech®.

Much of our time was spent grasping the differences between Windows and PM design, such as the following:

- Presentation Space versus Device Contexts
- Windows use of Short, and PM of Long for parameters
- PM and Windows reversing the top and bottom definition
- Bitmap handling (a difficult area)

Since Corel Draw is a native PM application, it runs with two companion routines called *Corel Trace* and *WFNBOSS*. Trace is a raster-to-vector conversion routine, and WFNBOSS is a font conversion utility that gives access to over 4,300 fonts from Adobe, Compugraphic®, Bitstream®, and ZSoft. Since these utility routines are compute-bound and non-graphics intensive, we ported them with the SMK. The Windows versions were relinked under OS/2. The SMK maps the Windows API calls to PM, allowing the utilities to run under PM with only a 10% performance degradation.

Connectivity

One design objective of Corel Draw was to make it as well connected as possible, since interoperability is critical to today's solutions. This allows users to maintain their investment in artwork created with drawing packages, such as MacDraw®, when they move to OS/2. A rich import selection also gives Corel Draw access to the clip art created in packages like Adobe Illustrator®. A menu of export file formats lets Corel Draw send graphics to a wide range of PC and Macintosh® publishing applications. Corel Draw supports the following file formats:

- Encapsulated PostScript (EPS)
- TIFF and PCX for bitmaps
- Computer Graphic Metafile (CGM)
- Picture Interchange Format (PIF)
- PM Metafile
- DXF (AutoCAD)
- Adobe Illustrator
- Macintosh Pict2

PM has proven to be an excellent development environment.

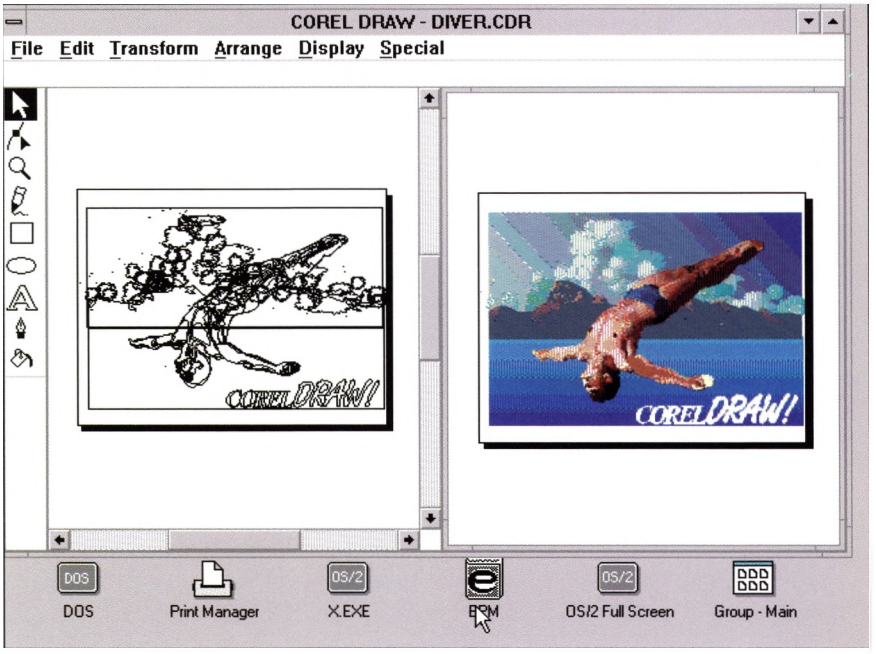


Figure 1. Example of Using Corel Draw to Render a Wire Frame

Corel Draw comes with over 100 vector scalable fonts.

Note the presence of PIF in this list. This is another unique feature of Corel Draw—supporting the import/export of GDDM (IBM mainframe and AS/400™ graphic access method) files. The PIF format is a subset of the one used by GDDM on IBM mainframes. As a PIF file is sent to the host, ProcessMaster or GDDM-based utilities transform it into a GDDM file. The GDDM file created by Corel Draw can then be placed into VM or MVS mainframe publishing applications, such as BookMaster®, or even displayed on an IBM graphic terminal, such as the 3179. This capability also can be used to place graphics in AS/400 publishing applications. Corel Draw also can import graphics produced in a PIF format, so any clip art existing in a GDDM format is accessible and revisable. The EPS filter can place Corel Draw graphics in host documents if the host is running the Adobe PostScript interpreter; the CGM filter can place a graphic into the IBM Computer-aided Acquisition Logistics (CALS) publishing applications for the U.S. Department of Defense.

The PM version of Corel Draw has added another filter for PM metafiles, which we parse and convert to our internal format. Corel Draw can import charts from packages like Microsoft Excel and Lotus 1-2-3/G for further modifications. The result can be saved as a Corel Draw file or as a file in any other format that we support. The metafiles can be read or written as external files or passed to the OS/2 Clipboard. Thus, Corel Draw might be described as a Systems Application Architecture (SAA) graphic editor.

Fonts

Corel Draw comes with over 100 vector scalable fonts. We also provide the WFNBOSS font conversion utility that gives us access to over 4,300 fonts from Adobe, Compugraphic, Bitstream, and ZSoft. These are converted into the internal Corel Draw font format. In this form, they can be used on any of the PM- or Windows-supported displays and printers. For example, if a user has licensed the Cheltenham font from Adobe, they can add it to their list of Corel Draw fonts.

THE PORTING PROCESS

Examples can best illustrate the similarities and differences between the Windows and OS/2 PM versions of Corel Draw. We used some new programming techniques to make programming in these two environments more consistent, while minimizing the effort of porting the application to other environments in the future. The following example shows how to write a single source code file that can generate a Windows or OS/2 PM application.

To begin, we need two constants for the #IF DEFINED argument, controlling compilation options under the Windows or OS/2 environments. Our goal was to eliminate these source code options as much as possible.

Two methods can define constants for the #IF DEFINED argument:

1. They can be defined in an include file.
2. The /D option can be used as the compiling option: cl /D.

The first method is preferable, and it would localize the constants in one file. Figure 2 shows an include file shell.hco that would be used in all the source files. In this example, the PM_SHELL is being selected.

Include files function better if they have a unique extension related to a given project.

```
#ifndef         SHELL_DOT_HCO           /* Test if already included */
#define         SHELL_DOT_HCO           /* no - include it */

//#define WIN_SHELL
#define PM_SHELL

#endif  /* SHELL_DOT_HCO */
```

Figure 1. C Header File (shell.hco) for Defining the "#if defined" Argument

For example, *.hco refers to the conversion include files. This solves a duplicate name problem. Having two statements at the start of every include file and one statement at the end prevents any recurring problem.

Throughout this article the following short formats are used:

- WndProc() refers to Window Processing routines.
- DlgProc() refers to Dialog Processing routines.

Getting Started

Typedefs are a fundamental technique used in writing a multi-environment application. The PM and Windows Software Development Kit (SDK) introduced many new typedefs, but we still needed to create additional ones to handle new environments. The first task, therefore, was to introduce some new typedefs. It is best to typedef as much as possible to a third general typedef, but sometimes users can still implement one from either environment. An include file defines these, as shown in Figure 3.

```
#ifndef CONVERT_DOT_HCO
#define CONVERT_DOT_HCO

#include "shell.hco"

//------------------------- WINDOW SHELL --------------------------
#if defined WIN_SHELL
#include <windows.h>
// ---- Message Argument Interface for windows and dialog handlers ----
typedef unsigned        M_MSG;
typedef WORD            M_MP1;
typedef LONG            M_MP2;
typedef BOOL            DLG_RETURN;
typedef LONG            WIN_RETURN;

// ---- window Processing return value ----
#define WIN_PROC_EXIT   FALSE;  // No more processing required
#define WIN_PROC_MORE   TRUE;   // More processing required

// ---- dialog Processing return value ----
#define DLG_PROC_EXIT   TRUE;   // No more processing required
#define DLG_PROC_MORE   FALSE;  // More processing required

// ---- Borrowed names from the OS/2 ----
#define DID_OK          IDOK
#define DID_CANCEL      IDCANCEL
#define WM_ERASEBACKGROUND WM_NULL

#define EXPENTRY pascal far _loadds
#define APIENTRY pascal far
typedef int   (pascal far  *PFN)();
typedef int   (pascal near *NPFN)();

typedef BYTE far * PBYTE;
typedef unsigned short USHORT;
typedef short SHORT;

#define VOID    void
typedef VOID    FAR *PVOID;

typedef struct _COMMANDMSG { /* commandmsg */
    USHORT  source;          /* mp2 */
    BOOL    fMouse;
    USHORT  cmd;             /* mp1 */
//    USHORT  unused;
} CMDMSG;
```

Figure 2. C Header File (convert.hco) for Conversion (Continued)

```
#define COMMANDMSG(pmsg) \
        ((struct _COMMANDMSG FAR *)     ( (PBYTE)pmsg  - sizeof(M_MP1) \
                                                       - sizeof(M_MP2) \
                                    ))

#endif

//------------------------- PM SHELL -------------------------
#if defined PM_SHELL
#include <os2.h>
// ---- Message Argument Interface for windows and dialog handlers ----
typedef USHORT          M_MSG;
typedef MPARAM          M_MP1;
typedef MPARAM          M_MP2;
typedef MRESULT         DLG_RETURN;
typedef MRESULT         WIN_RETURN;

// ---- window Processing return value ----
#define WIN_PROC_EXIT   FALSE;  // No more processing required
#define WIN_PROC_MORE   TRUE;   // More processing required

// ---- dialog Processing return value ----
#define DLG_PROC_EXIT   FALSE;   // No more processing required
#define DLG_PROC_MORE   TRUE;    // More processing required

#endif
#endif  /* CONVERT_DOT_HCO */
```

Figure 2. C Header File (convert.hco) for Conversion

WndProc()

Figures 4 and 5 show a typical WndProc() in the Windows and PM environments. Their differences start from line one, the function prototype:

- The return value is BOOL under WIN and MRESULT under PM.
- The arguments are different types.

Using the typedefs, Figure 6 shows a new style for the function prototype that can be used in both environments.

The next difference is in the WM_COMMAND case handling. PM offers a few stack macros to extract message data. These macros can extract keyboard, mouse, and command data. There is a window version of this command macro in convert.hco. Figure 6 shows what the new version looks like.

Menu Message

Upon receiving the IDM_ABOUT id, the program calls the AboutDlgProc(). Here is a case where we would write a general purpose routine to be compiled differently under a given environment. The objective is to pass all the required arguments for both environments, though not all would be used for a given environment. Using a second macro shown in Figure 7, the required arguments would be filtered out and passed to a more environment-specific function shown in Figure 8.

Under PM, the frame window sends the WM_ERASEBACKGROUND message to a client window when the background is to be redrawn. If an application returns DLG_PROC_MORE, the frame-window procedure erases the rectangle of the frame window covered by the client window. It does this by filling it with the system color.

```
#include <windows.h>
#include "sample.h"     /* specific to this program      */
long FAR PASCAL MainWndProc(hWnd, message, wParam, lParam)
HWND hWnd;          /* window handle                     */
unsigned message;   /* type of message                   */
WORD wParam;        /* additional information            */
LONG lParam;        /* additional information            */
{
    FARPROC lpProcAbout;        /* pointer to the "About" function */
    switch (message) {
        case WM_COMMAND:        /* message: command from application menu */
            if (wParam == IDM_ABOUT) {
             lpProcAbout = MakeProcInstance(About, hInst);
             DialogBox(hInst,  /* current instance         */
                 "AboutBox",   /* resource to use          */
                 hWnd,         /* parent handle            */
                 lpProcAbout); /* About() instance address */
             FreeProcInstance(lpProcAbout);
             break;
            }
            else                    /* Lets Windows process it       */
             return (DefWindowProc(hWnd, message, wParam, lParam));
        default:                    /* Passes it on if unprocessed    */
            return (DefWindowProc(hWnd, message, wParam, lParam));
    }
    return (NULL);
}
BOOL FAR PASCAL About(hDlg, message, wParam, lParam)
HWND hDlg;              /* window handle of the dialog box */
unsigned message;       /* type of message                 */
WORD wParam;            /* message-specific information    */
LONG lParam;
{
    switch (message){
        case WM_COMMAND:        /* message: received a command */
            if (wParam == IDOK  /* "OK" box selected?          */
             || wParam == IDCANCEL) {  /* System menu close command? */
             EndDialog(hDlg, TRUE);    /* Exits the dialog box        */
             break;
            }
            break;
        default:
             return FALSE;
    }
    return TRUE;        /*  If it has processed a particular message. */
}
```

Figure 3. Typical Windows WndProc()

A similar message under Windows is WM_ERASEBKGND, but it is handled differently. To solve this problem, WM_ERASEBACKGROUND is defined as WM_NULL under the WIN_SHELL.

In the case of default handling, it was necessary to choose between DefWindowProc() in Windows and WinDefWindowProc() in PM. The latter one was chosen (shown in Figure 6).

If the function is handling any message, WIN_PROC_EXIT is returned. This eliminates a variant return value for a specific environment. Note that a similar define is required in DlgProc() handling routines, since the return value is different in the two environments.

DlgProc()

Comparing the DlgProc() routines in Figure 4 and 4, the first difference in the function prototype can be solved as it was in the WndProc() function. The WM_COMMAND case also is handled in a similar fashion.

```
#define INCL_WIN
#define INCL_GPI
#include <os2.h>
#include "sample.h"     /* specific to this program      */

MRESULT EXPENTRY MainWndProc( hwnd, msg, mp1, mp2 )
HWND    hwnd;
USHORT  msg;
MPARAM  mp1;
MPARAM  mp2;
{
    switch( msg ){
        case WM_COMMAND:        /* message: command from application menu */
           switch( SHORT1FROMMP( mp1 ) ){
               case IDM_ABOUT:
                WinDlgBox(
                    HWND_DESKTOP,   /* Handle of the parent */
                    hwnd,           /* Handle of the Owner  */
                    AboutDlgProc,   /* Dialog proc address  */
                    NULL,           /* handle of resource   */
                    IDDLG_ABOUT,    /* id of the resource   */
                    NULL            /* address of Data      */
                );
                break;
             default:
                    /* Let window process it       */
                    return WinDefWindowProc(hwnd, msg, mp1, mp2);
           }
           break;
        default:    /* Passes it on if unprocessed    */
            return WinDefWindowProc(hwnd, msg, mp1, mp2);
    }
    /*  If it has processed a particular message. */
    return NULL;
}
MRESULT EXPENTRY AboutDlgProc(hDlg, msg, mp1, mp2)
/*
    About... dialog procedure
*/
HWND    hDlg;
USHORT  msg;
MPARAM  mp1;
MPARAM  mp2;
{
    switch( msg){
        case WM_COMMAND:        /* message: received a command */
             switch(COMMANDMSG(&msg)->cmd)
             {
                case DID_OK:         /* "OK" box selected?         */
                case DID_CANCEL:     /* System menu close command? */
                    WinDismissDlg(hDlg, TRUE);
                    break;
             }
             break;
        default:
             return WinDefDlgProc( hwndDlg, msg, mp1, mp2);
    }
    /*  If it has processed a particular message. */
    return FALSE;
}
```

Figure 4. Typical PM WndProc()

```
/* wndproc.c - C Source Code Listing for sample program */

#define INCL_WIN
#define INCL_GPI

#include "convert.hco"  /* Conversion  file              */
#include "dlg.hco"      /* Dialog file                   */
#include "sample.h"     /* specific to this program      */

/*
------------------------------------------------------------------------------
WIN_RETURN APIENTRY MainWndProc( hwnd, msg, mp1, mp2)
HWND  hwnd;      /- window handle                    -/
M_MSG msg;       /- type of message                  -/
M_MP1 mp1;       /- additional information           -/
M_MP2 mp2;       /- additional information           -/

The MainWndProc function processes the following messages from the app thread's
message queue.

Comments
        To process the IDM_ABOUT message, call  CorProcessDlg().  This would
        isolate the handling of the dialog boxes under a given environment.
------------------------------------------------------------------------------
*/
WIN_RETURN APIENTRY MainWndProc(HWND hwnd, M_MSG msg, M_MP1 mp1, M_MP2 mp2 )
{

    switch( msg ){
        case WM_COMMAND:           /* message: command from application menu */
           switch( COMMANDMSG( &msg ) -> cmd ){
               case IDM_ABOUT:
                  WinDlgBox(
                    HWND_DESKTOP,      // handle of the parent window
                    hwnd,              // handle of the owner window
                    (PFN)AboutDlgProc, // dialog procedure address
                    ghInstance,        // handle of resource or the instance
                    IDDLG_ABOUT,       // identifies the dialog
                    NULL               // address of application-specific data
                    );
                  break;
             default:
                    /* Let window process it       */
                    return WinDefWindowProc(hwnd, msg, mp1, mp2);
           }
           break;

        case WM_ERASEBACKGROUND:
             return WIN_PROC_MORE;
        default:    /* Passes it on if unprocessed    */
            return WinDefWindowProc(hwnd, msg, mp1, mp2);
    }

    /*  If it has processed a particular message. */
    return WIN_PROC_EXIT;
}
```

Figure 5. Multi-Environment WndProc() (Continued)

```
/*
------------------------------------------------------------------------------
DLG_RETURN EXPENTRY AboutDlgProc( hwndDlg, msg, mp1, mp2)
HWND  hwndDlg;  /- window handle                    -/
M_MSG msg;      /- type of message                   -/
M_MP1 mp1;      /- additional information            -/
M_MP2 mp2;      /- additional information            -/

It processes the messages for "About" dialog box.
Comments
        Wait for user to click on "OK" button, then close the dialog box.
------------------------------------------------------------------------------
*/
DLG_RETURN EXPENTRY AboutDlgProc(HWND hwndDlg, M_MSG msg, M_MP1 mp1, M_MP2 mp2
)
{
    switch( msg ){
        case WM_COMMAND:            /* message: received a command */
            switch( COMMANDMSG( &msg )-> cmd ){
             case DID_OK:      /* "OK" box selected?          */
             case DID_CANCEL:  /* System menu close command? */
                    /* Exits the dialog box        */
                    WinDismissDlg( hwndDlg, TRUE );
                    return DLG_PROC_EXIT;
            }
            break;
        default:
             return WinDefDlgProc( hwndDlg, msg, mp1, mp2);
    }

    /*  If it has processed a particular message. */
    return DLG_PROC_EXIT;
}
```

Figure 5. Multi-Environment WndProc()

```
#ifndef         DLG_DOT_HCO             /* Test if already included */
#define         DLG_DOT_HCO             /* no - include it */

//------------------------- WINDOW SHELL --------------------------
#if defined WIN_SHELL

#define HWND_DESKTOP    1;

DLG_RETURN CorDlgBoxWIN( HWND hwndOwner, PFN pfn, HANDLE hInstance, SHORT idDlg );
#define WinDlgBox(    hwndParent, hwndOwner, pfn, hInstance, idDlg, pCreateParams ) \
        CorDlgBoxWIN(             hwndOwner, pfn, hInstance, idDlg )

#define WinDismissDlg( hwndDlg, usResult) \
        EndDialog( hwndDlg , usResult )

#define WinDefWindowProc( hwnd, msg, mp1, mp2 ) \
        DefWindowProc( hwnd, msg, mp1, mp2 )

#define WinDefDlgProc( hwnd, msg, mp1, mp2 )    FALSE
#endif

#endif  /* DLG_DOT_HCO */
```

Figure 6. C Header File (dlg.hco) for Dialog Box Routines

```
#define INCL_WIN
#define INCL_GPI
#include "convert.hco"  /* Conversion  file             */
#include "dlg.hco"      /* Dialog file                  */
#include "sample.h"     /* specific to this program     */

#if defined WIN_SHELL
DLG_RETURN CorDlgBoxWIN( HWND hwndOwner, PFN pfn, HANDLE hInstance,
                         SHORT idDlg )
{
    BOOL    fRet;
        pfn = MakeProcInstance( pfn, hInstance);
        fRet = DialogBox(
             hInstance,              /* current instance         */
             MAKEINTRESOURCE(idDlg), /* resource to use          */
             hwndOwner,              /* Owner handle             */
             pfn                     /* About() instance address */
             );
        FreeProcInstance( pfn );
        return fRet;
}
#endif
```

Figure 7. Dialog Box Handling Routine (dlgproc.c)

The dismissing function name of the dialog box is again chosen from the PM SDK and defined in the convert.hco.

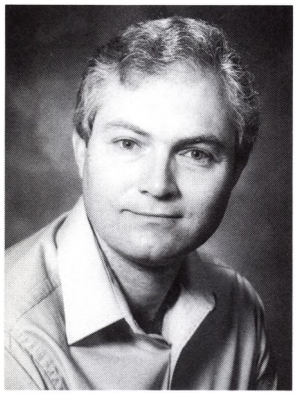
Windows and PM process default handling of the dialog boxes differently. PM uses a new function called WinDefDlgProc(). Our sample program uses this function; the equivalent macro is in dlg.hco.

John McFetridge, *Corel Systems Corporation, 1600 Carling Avenue, Ottawa, Ontario K1Z 8R7, Canada. (613) 728-8200. Mr. McFetridge has been director of OS/2 development and special projects for Corel since December 1988. He was founder and president of Simware Inc., a mainframe communications company. He authored SIM3278, the communication package on which Simware® is based. Previously, he had five years experience as a VM/CMS systems programmer. He has a degree in Mathematics from Dalhousie University in Halifax, Nova Scotia.*

Razmik Malek-Adamian, *Corel Systems Corporation, 1600 Carling Avenue, Ottawa, Ontario, K1Z 8R7, Canada. (613) 728-8200. Mr. Malek-Adamian was the lead software engineer for the OS/2 version of Corel Draw. His career has spanned many aspects of computing including software programming and hardware design. He has a degree in Electrical Engineering from Carleton University in Ottawa.*

Multimedia & Graphics

Collage PM: A Raster (Bitmap) Graphics OS/2 PM Application



Glen R. Horton, Sr.

Collage PM can capture OS/2 PM screens, and display and print raster images of many types.

by Glen R. Horton, Sr.

This article describes some tips and techniques for creating OS/2 Presentation Manager (PM) bitmap graphics applications. It discusses bitmapped image manipulation using both OS/2 PM Gpi and Dev functions. Depending on the level of required output and performance, the designer must decide whether to use standard Gpi functions or augment them by adding a few more lines of code to increase the perceived performance of the application.

Collage PM™ is a multithreaded OS/2 PM application that makes extensive use of the standard OS/2 PM Application Programming Interfaces (APIs) for screen and printer output of bitmapped (raster) images. This article is based on some design and implementation techniques used during Collage PM's development. Collage PM can capture OS/2 PM screens, and display and print raster images of many types. It performs image conversions, pixel-level cropping, and transformations among industry standard raster image file formats (PCX, TIFF, clipboard, BMP, and metafile). Collage PM also allows the user to rename, delete, and copy raster image files.

Figure 1 illustrates a typical image viewing operation available with Collage PM. An Image Management dialog box provides the principal user interface for selecting the Collage PM functions. A view window shows the image reduced to fit within its client area. Additionally, a magnifier window displays a 1:1 view of the image area outlined by the dotted-line rectangle (viewport) shown in the view window. The viewport automatically tracks the size of the magnifier window. It can be moved within the reduced image by using the mouse, scroll bars, or keyboard arrow keys.

Collage PM offers screen image capturing, with a valuable preview capability. Previewing offers all the capabilities previously described for image viewing under Collage PM. A settable countdown timer is provided to allow enough time for the user to organize the OS/2 screen before it is captured. The OS/2 screen resolution and color capabilities are automatically determined by Collage PM; therefore, no preconfiguration or special drivers are needed.

Images can be cropped and printed by Collage PM. It supports any standard OS/2 PM printer capable of printing raster graphics. A print control window allows the user to control the image's aspect ratio, sizing constraints, and position on the printed page. An image can be converted and transformed as it is printed by selecting the appropriate settings in the Image Output Settings dialog box.

USING GPIBITBLT FOR SCALING BITMAPS

The standard GpiBitBlt function provides a powerful, general purpose method for variably sizing a bitmapped image. A common practice when scaling a bitmap is to supply a set of points describing the source and target rectangles as an argument to GpiBitBlt. The relationship between these rectangles defines the scaling and position of the resulting bitmap in the target space.

Reducing an image initially results in some loss of image detail, as some information will be discarded or somehow combined. To help minimize the effects of this loss, the application programmer can specify how the in-between pixels will be treated. The options available are to OR, AND, or IGNORE these in-between pixels.

In general, to retain as much image detail as possible when reducing monochrome images, use the OR option for mostly black images and the AND option for mostly white images. The IGNORE option is useful when reducing simple (non-dithered) monochrome or color bitmaps, but there will be a loss of image detail as the reduction factor increases.

For example, if a 100-scan-line image is reduced by a factor of ten in the vertical direction, then every tenth scan line will be sampled when the IGNORE option is specified. This sampling results in discarding nine out of every ten scan lines.

With a source image that is mainly composed of horizontal lines, a good choice would be to AND or OR the extra scan lines depending on the color of the background pixels. Conversely, using the IGNORE option in this case may result in an image containing only the background or foreground color with a complete loss of image detail.

Images containing repeating patterns are particularly susceptible to scaling side effects because the patterns in the image may change as in-between scan lines and pixels are either combined or ignored as the scaling occurs. In Figure 2, a 4x4 50% white pattern with alternating black and white pixels is reduced horizontally by a factor of two. The resulting pattern has changed into a 4x2 50% white pattern that has horizontal stripes instead of a more diffuse distribution of pixels. If the original 4x4 pattern is reduced by a factor of two in both the horizontal and vertical direction, the resulting pattern will depend on the horizontal and vertical sampling order. For example, if row 1 of the source pattern is sampled first and then row 3 is sampled, the resulting pattern would be all black.

The same principles apply to dithered color images. Color changes can occur as in-between data in the dithered patterns is combined or ignored during the scaling operation. For best enlargement of a dithered color or dithered monochrome image, equal horizontal and vertical integer scalings should be used. Equal integer scaling in both axes will leave the dithered patterns intact. Of course, there is a practical limit to the enlargement, since the patterns will eventually become too large to produce a dithered color at any normal viewing distance.

Choosing the appropriate option (OR, AND, IGNORE) for any given image affects how universal a program will be for a range of input image types. The programmer may make fixed assumptions about the likely images, or may choose to include some algorithm that first determines what type of image is being scaled, and then applies the correct combining option.

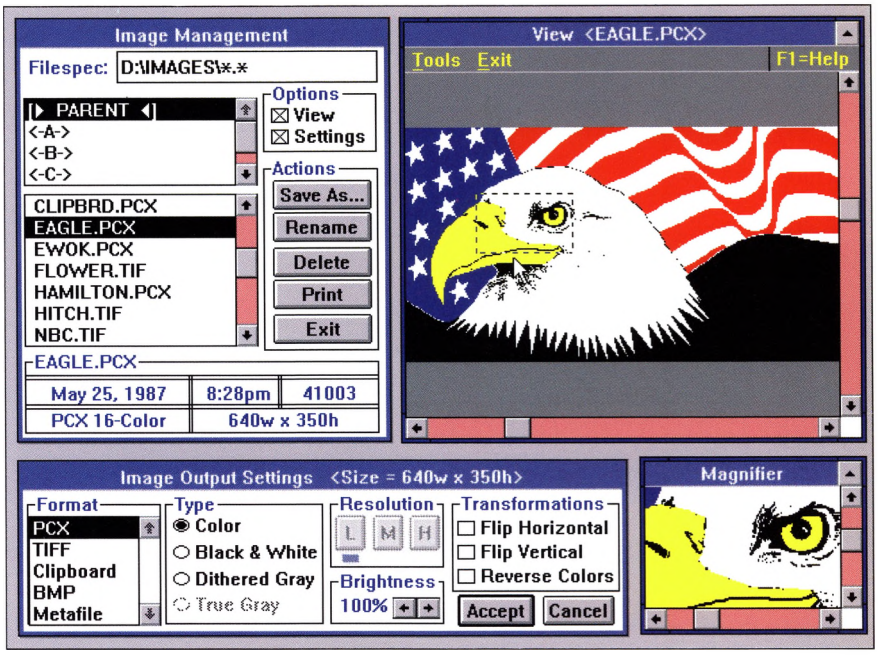


Figure 1. A Typical Image Viewing Operation

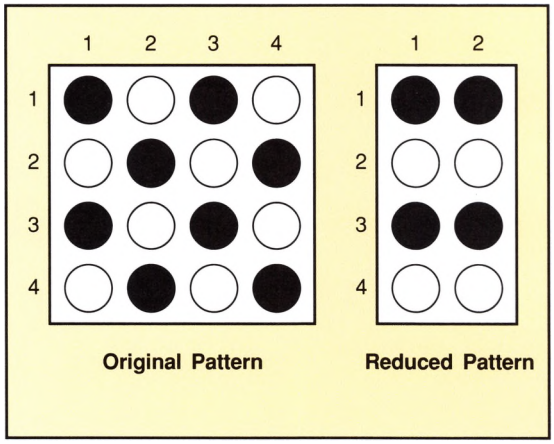


Figure 2. The Effects of Horizontal Reduction on Dithered Monochrome or Color Images

WHERE GPIBITBLT CAN BE ENHANCED

Using GpiBitBlt to size images can result in undesirable side effects. Examples of these effects include:

- **Moire Patterns and Checkerboarding**: Moire patterns in scaled bitmapped images usually appear at regular horizontal and vertical intervals. In color images, moire patterns may be seen as a change or loss of color in repetitive horizontal or vertical bands. In monochrome images, they are normally seen as a blank space or loss of image definition.

  Applying vertical and horizontal scaling simultaneously may produce a checkerboard effect in the scaled image. A checkerboard pattern is the result of mixing the horizontal and vertical moire patterns. The severity of these side effects relates to the image content, color, and scaling factors. Figure 3 illustrates the moire pattern and checkerboard side effects that result from a normal GpiBitBlt operation.

- **Image Truncation**: As the vertical dimension of a bitmapped image increases, a subtle error in the OS/2 PM vertical scaling algorithm manifests itself as image truncation. As the vertical size grows, the amount of truncation increases for a fixed vertical reduction. As the vertical reduction increases for a given image, more scan lines at the top of the image are lost. This vertical truncation only occurs when the vertical dimension of the image exceeds the vertical resolution of the graphics display. Figure 4 illustrates the truncation side effect.

Although the standard GpiBitBlt call sometimes produces moire patterns in both vertical and horizontal axes, it can produce image truncation in the vertical axis only. The moire patterns represent a relatively minor visual degradation of image quality. Image truncation, however, is usually unacceptable, since it can eliminate from view a significant portion of the image. It also can wreak havoc with the accurate display of cropping rectangles and viewports that are supposed to align with specific details in a reduced image.

Certain video drivers may cause this image truncation—both EGA and VGA drivers have been observed to exhibit this problem. Regardless of the cause, the programmer must find a suitable way to maintain the visual quality of reduced images.

One method that provides an acceptable tradeoff between output quality and processing speed is to call GpiBitBlt multiple times, discarding intermediate scan lines along the way. This method eliminates vertical image truncation and also eliminates moire patterns because of vertical sizing.

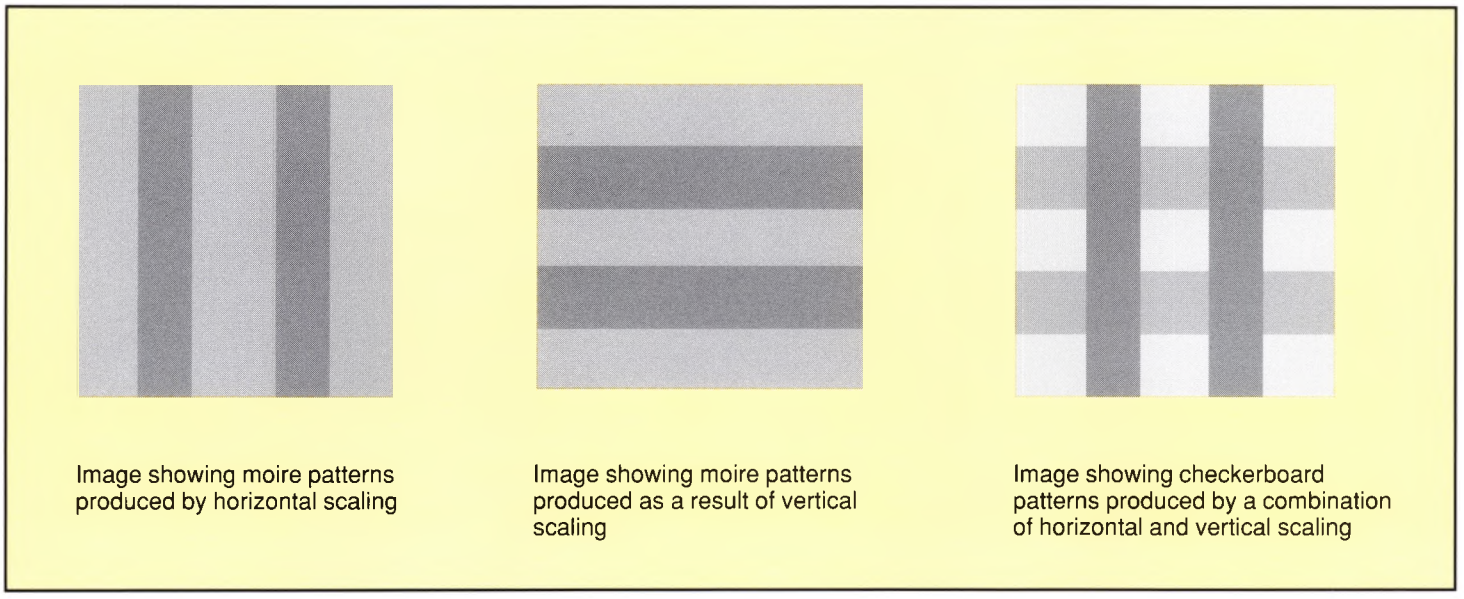


Image showing moire patterns produced by horizontal scaling
Image showing moire patterns produced as a result of vertical scaling
Image showing checkerboard patterns produced by a combination of horizontal and vertical scaling

Figure 3. Moire Pattern and Checkerboard Effects

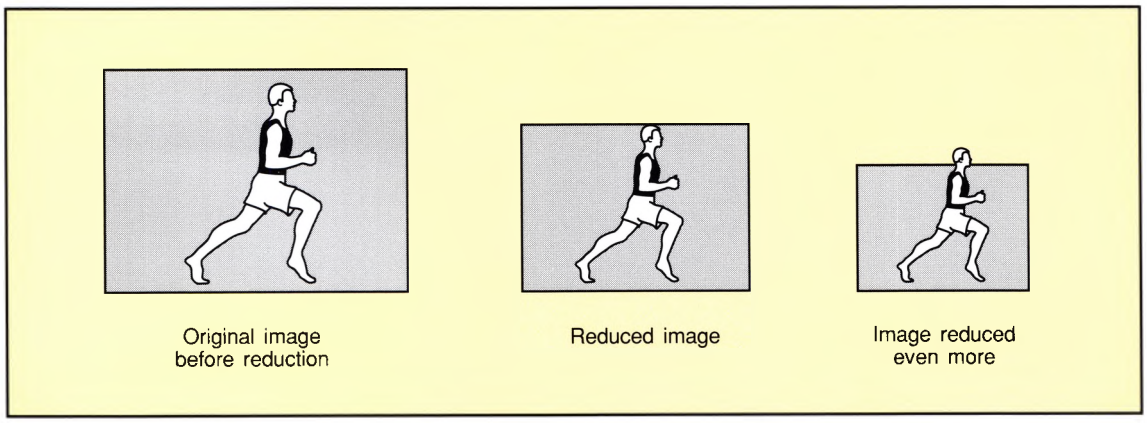


Figure 4. Image Truncation Side Effects Resulting from GpiBitBlt Reduction

Figure 5 shows an example of this method. In this example, the number of calls is calculated by dividing the image height by the desired vertical reduction factor. The points describing the source and target rectangles are calculated for each call to GpiBitBlt.

For this method, the source and target rectangles each have a height of one scan line. The source rectangle has a width equal to the image width. The target rectangle has a width that is calculated to produce the desired horizontal scaling. GpiBitBlt performs the horizontal scaling for the selected scan line and transfers it to the target bitmap. It then calculates the indices to the next source and target scan lines. The process repeats until it scales the last source scan line.

```
BOOL ReduceBitmap(hpsDest, hpsSource, pbmiSource, sHRfactor, sVRfactor)
HPS       hpsDest;                      // handle to Destination PS
HPS       hpsSource;                    // Handle to Source PS
PBITMAPINFO  pbmiSource;                // Bitmapinfo header for size etc
SHORT     sHRfactor                     // Horizontal reduction factor
SHORT     sVRfactor;                    // Vertical reduction factor

{
POINTL    ptl[4];
LONG      lrc;
//  hpsSource  and hpsDest already have bitmaps set with correct
//    sizes and colors
//  pbmiSource is already filled in by calling routine
//  sHRfactor and sVRfactor specify the horizontal and
//    vertical reduction factors respectively

  // Initializations
  // Target Rectangle is defined by 2 points
  ptl[0].x = 0;                                     // Lower Left x
  ptl[0].y = 0;                                     // Lower Left y
  ptl[1].x = (LONG)(pbmiSource -> cx / sHRfactor);  // Upper Right x
  ptl[1].y = ptl[0].y + 1;                          // Upper Right y

  // Source Rectangle is defined by 2 points
  ptl[2].x = 0;                                     // Lower Left x
  ptl[2].y = 0;                                     // Lower Left y
  ptl[3].x = (LONG)(pbmiSource -> cx);              // Upper Right x
  ptl[3].y = ptl[2].y + 1;                          // Upper Right y

  lrc = GPI_OK;   // assume that the return is ok for now
```

Figure 5. One Method to Eliminate Image Truncation and Minimize Moire Patterns (Continued)

```
  while((ptl[2].y < pbmiSource -> cy) && (lrc == GPI_OK)){
    // Scale source scan line horizontally by GpiBitBlt function
    lrc = GpiBitBlt(hpsDest, hpsSource, 4L, ptl, ROP_SRCCOPY, BBO_IGNORE);

    // calculate new index of next target scan line
    ptl[0].y += 1;
    ptl[1].y += 1;

    // calculate new index of next source scan line
    ptl[2].y += (LONG)(sVRfactor);
    ptl[3].y += (LONG)(sVRfactor);
  }
  // return to caller with any error that may have occurred
  return(lrc)
}
```

Figure 5. One Method to Eliminate Image Truncation and Minimize Moire Patterns

Multiple asynchronous threads are created to isolate the main user interface thread from the process-intensive tasks.

This technique is applicable universally to both monochrome and color bitmaps. The algorithm is easily adapted to non-integer reductions by changing the source and target scan index calculations, and then rounding the results to their nearest whole value. The source and target rectangles should be within the bounds of their respective bitmaps.

Another adaptation to this method is to increase the height of the source rectangle to equal the reduction factor and leave the height of the target rectangle set to 1. Then, the GpiBitBlt OR and AND options could be used to adjust the output, as described earlier.

This method does not remove the moire patterns that appear during horizontal scaling. To remove these patterns requires application or Dynamic Link Library (DLL) code that correctly handles all of the standard cases for monochrome and color bitmaps. Since OS/2 PM raster-type device drivers are required to handle bitmaps of 1, 2, 4, 8, and 24 bits per pixel, this development effort is significant. It would require constant upgrading as bitmapped standards evolve.

Unlike vertical image truncation, moire patterns that result from horizontal scaling do not typically obscure or eliminate meaningful amounts of image detail. A reasonable design choice might be to ignore the effect, and continue to use the standard GpiBitBlt function. This decision, however, must be weighed against the degree of image quality required.

ORCHESTRATING THE USE OF TIME-CONSUMING GPI FUNCTIONS

Since bitmap files are not compressed, they can become very large. For example, a simple 8.5 x 11 inch black and white image scanned at 300 dpi can result in a raw, uncompressed file larger than 1 MB. To achieve acceptable display and image manipulation performance, time-consuming operations like bitmap scaling must be performed in a way that does not cause the user or other applications to wait. This section describes some techniques to help increase the performance.

Many factors and personal viewpoints affect the user's perception of an application's performance. OS/2 keeps the user busy and, at the same time, maximizes the use of system hardware and software resources. Applications that handle bitmapped images frequently need to perform time-consuming operations, such as image scaling and rotating. With OS/2, these applications can remain responsive to the user's inputs by spawning additional threads of execution that handle process-intensive tasks in the background.

In addition, the user should have the option to interrupt an operation at any time. This requires the main user interface thread of an application to monitor and communicate with other threads that have been created. In Collage PM, multiple asynchronous threads are created to isolate the main user interface thread from the process-intensive tasks that are being performed. Creating and monitoring multiple threads is only part of the solu-

tion, since at some point the system must perform the work.

Collage PM has a technique that helps to smooth out the waiting periods and increase the overall performance. Collage PM keeps internal records of bitmap scaling, transformations, and the current bitmap's relationship to the original image information from which it was created. Then, Collage PM only performs the transformations and scaling that are absolutely necessary to update what the user sees on the screen.

This approach is used to make intelligent choices about when to perform image resizing operations. For instance, if the user changes the size of a view window, Collage PM analyzes the current bitmap condition and determines if a new reduction is necessary. If the reduction is not needed, the existing bitmap is simply centered within the new client space. The time-consuming reduction process is completely bypassed. Other time-saving techniques may include preserving one or more bitmapped images during conversions, or briefly maximizing and then restoring the view windows. It is wise to use this particular method sparingly to avoid monopolizing precious system resources.

BITMAPS AND DEVICE COLORS

A bitmap set to a Presentation Space (PS) that is associated with a Memory Device Context (DC) usually has a limited color scope. It is necessary to perform true color image manipulations outside of this context. Using the color values returned by GpiQueryBitmapBits can produce undesirable results. This section discusses this limitation and suggests an alternative strategy.

When DevOpenDC creates a DC, the user must specify a compatible DC, or the DC is automatically defaulted to the screen DC. A bitmap set to a PS associated with a DC (one that is compatible with the screen) is color-adjusted to match the display device's capabilities and current color set. Before the bitmap bits can be retrieved by GpiQueryBitmapBits, the bitmap must be set to a PS. This requirement limits which colors the Gpi functions can display and retrieve. Figure 6 illustrates how the color data changes when a bitmap is set to a PS.

Operations on image data must be performed independently from the bitmap that is displayed. That is because the color information is adjusted automatically. When the bitmap data is retrieved after the bitmap has been set to a PS, a loss of color definition will probably occur. A good example is trying to display a 256-color image as a bitmap on a PM display that supports only 16 concurrent colors.

The color data in the bitmap will automatically be adjusted to fit the current 16 colors that the associated PM driver set for the display. OS/2 PM determines the best fit to the current color set for the DC and associated PS. The displayed bitmap will exhibit a high contrast; in some cases, it will lose much of its detail.

The strategy used in Collage PM allows the user to display an image or print to the limit possible with a particular display or printer device and its associated PM driver. Yet, it retains its original color content when converting that image to another compatible format. This strategy maintains complete image data integrity by always referring to the original source image, rather than using an intermediate and possible degraded (lossy) stage as input to the conversion process.

For example, when Collage PM converts a color image to a dithered monochrome image, the original image data is used for the conversion. The resulting data can create the final output bitmap. OS/2 PM and its device drivers perform automatic color adjustments that completely insulate conversion and transformation operations.

Collage PM only performs the transformations and scaling that are absolutely necessary to update what the user sees on the screen.

CHECKING THE ERROR RETURNS FROM GPI AND DEV FUNCTIONS

Potentially fatal errors can be encountered when making calls to Gpi and Dev functions. Depleting or misusing OS/2 PM resources, such as Memory Device Contexts, and Presentation Spaces, usually causes these errors. They can also occur when a PM device driver fails. Handling these errors requires careful monitoring, reporting, and processing of the error codes returned by the Gpi and Dev functions. For example, neglecting to adequately maintain active DCs, PSs, and bitmaps can result in a system trap error, increasingly slow system performance, or in some extreme cases, loss of the desktop's

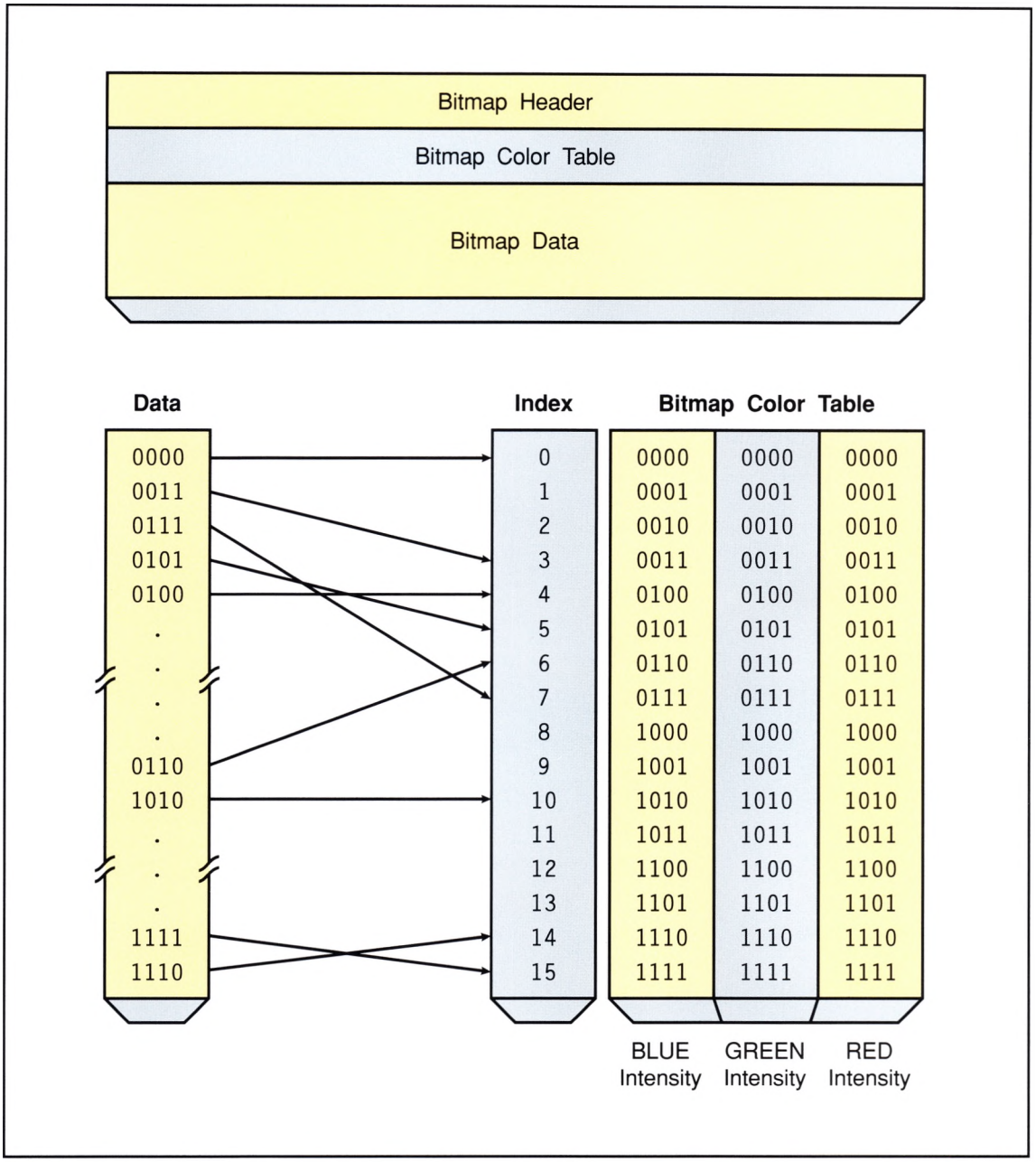


Figure 6. Bitmap Color Mapping

visual integrity. System trap errors may even force the user to abort the program, possibly losing valuable data and time. This section discusses a safe exit routine to help in the event of a fatal error.

When DCs, PSs, and bitmaps are created and subsequently associated with one another, an orderly exit procedure must be followed to systematically and completely close and destroy them. This presents a problem when the error condition is critical enough to abort the current operation, but the code is deeply nested. A strategy used in Collage PM to ensure a safe exit requires a separate exit routine that automatically determines whether a DC, PS, or bitmap exists.

After determining the cause of the error, a call is made to the safe exit routine. Figure 7 shows an example of the routine that can be used to exit safely from Gpi and Dev error conditions when DCs, PSs, and bitmaps have been created. This routine performs all neces-

sary steps to close and destroy the DCs, PSs, and bitmaps. It can be called from any point within code that may be using the DCs, PSs, and bitmaps.

The safe exit routine relies on global variables containing handles to the currently active DCs, PSs, and bitmaps. If a handle to one of these elements is NON-NULL, it is assumed to exist, and the proper close or destroy operation must be performed. After the close or destroy is successful, the affected handle variable will be set to NULL.

The exit routine returns a TRUE if the exit operation was safe, and a FALSE if an error was detected during the close or destroy operation. If the exit routine itself returns an error, it is best to terminate the application quickly.

It is helpful to inform the user of fatal errors and ask for an acknowledgement before actually terminating the operation. In some cases it may be appropriate to retry an operation, such as when a printer has a correctable problem (for example, out of paper). A simple fatal error routine can be constructed to display a one- or two-line message to the user and ask for acknowledgement. Displaying the cause of fatal errors also helps technical support personnel to identify quickly the cause of a problem when a user calls for assistance.

If a fatal error affects the system as a whole (such as when the system has run out of memory), it is acceptable to display a System Modal message box. This message box requires that the user acknowledge it before continuing. If the error is local to the application, it is inappropriate to display a System Modal message box. An Application Modal message box that suspends only the application displaying the message and permits other applications to continue is normally used.

```
// Global Variables containing the handles to the DC, PS, and bitmap
// to be closed and destroyed
HDC     hdc;
HPS     hps;
HBITMAP hbm;
HMF     hmf;

BOOL SafeExit()
// Returns TRUE if everything is OK
// Returns FALSE if a problem was encountered while cleaning up
{
// Temporary variable for holding bitmap handle
HBITMAP  hbmOld;
HMF      hmfReturn;
BOOL     frc;

// Assume everything will happen ok
frc = TRUE;
// Assume that we are not closing a DC associated with METAFILE
hmf = (HMF)NULL;

// Delete any bitmap that was created
  if(hbm != (HBITMAP)NULL){
    // The bitmap has been successfully created before reaching this routine
    // Get the handle to the bitmap currently set to this PS
    // Release the Bitmap from the PS
    hbmOld = GpiSetBitmap(hps, NULL);
    if(hbmOld != (HBITMAP)NULL){
      // Delete the bitmap and release memory used by bitmap
      if(GpiDeleteBitmap(hbmOld)){
        // Reset the BITMAP handle variable to unused
        hbm = (HBITMAP)NULL;
      }
      else{
        // Encountered an error while deleting bitmap
        frc = FALSE;
      }
```

Figure 7. Example of Routine to Exit from Gpi and Dev Error Conditions (Continued)

```
    }
  }

  if(frc){
    // Destroy any PS that was created
    if(hps != (HPS)NULL){
      // Break the association of the PS with its associated DC
      GpiAssociate(hps, NULL);
      if(GpiDestroyPS(hps)){
        // reset the PS handle variable to unused
        hps = (HPS)NULL;
      }
      else{
        // Encountered an error while destroying PS
        frc = FALSE;
      }
    }
  }

  if(frc){
    // Close any DC (or get handle to METAFILE) that was opened
    if(hdc != (HDC)NULL){
      hmfReturn = DevCloseDC(hdc);
      if(hmfReturn == DEV_OK){
        hdc = (HDC)NULL;
      }
      else{
        if(hmfReturn == DEV_ERROR){
          // Encountered an error while closing DC
          frc = FALSE;
        }
        else{
          // Just closed a metafile DC
          hdc = (HDC)NULL;
          // Applications using metafiles may want to return the hmfReturn
          // value to the calling routine.
          hmf = hmfReturn;
        }
      }
    }
  }
  return(frc);
}
```

Figure 7. Example of Routine to Exit from Gpi and Dev Error Conditions

OS/2 PM DEVICE DRIVERS MAY LACK SUPPORT FOR GPIBITBLT

Users should always query the device's capabilities *before* attempting any output to the device. Calling an OS/2 PM driver to perform GpiBitBlt operations that it does not support will cause unpredictable results. The user may have to terminate the application prematurely.

OS/2 PM does support output devices that are incapable of supporting raster (bitmap) output, such as vector devices like plotters. Most printers have hardware support for raster output, but OS/2 PM does not require that the drivers provide access to this capability. The OS/2 PM screen device drivers, however, must support raster operations. In some cases, the same BitBlt code used in the screen device driver is used by a printer device driver to simulate GpiBitBlt operations. This is normally done to save driver code and to produce raster output for other raster-capable devices that is consistent with bitmap output to the screen.

In general, an OS/2 PM application should determine the range of output capabilities for a device. If a device does not provide support for the capabilities necessary to achieve successful output, the user should be informed. Once the user acknowledges the incompatibility, an alternate output device that meets the requirements can be easily selected or installed. An example of querying the default printer's device capabilities is shown in Figure 8.

```
// EXTERNAL GLOBAL variables used by GetDefaultPrinter()

UCHAR szLogPrinterAddr[10] = {0};
UCHAR szPrinterDvrName[32] = {0};
UCHAR szPrinterDataType[32] = {0};
LONG  lPrinterDevType;
LONG  lDevCaps[CAPS_VERTICAL_FONT_RES + 1];
BOOL  RasterOk;

DEVOPENSTRUC dopPrinter = {szLogPrinterAddr, szPrinterDvrName,
                           NULL, szPrinterDataType, (PSZ)NULL,
                           "PMPRINT", (PSZ)NULL, (PSZ)NULL,
                           (PSZ)NULL};

BOOL GetDefaultPrinter(habCurrent)
HAB     habCurrent;
{
char    szKeyname[80];
char    szDetails[256], szTempString[256];
PCH     pchScan;
HPS     hpsError;
POINTL  ptlError;
UCHAR   ErrorString[128];
HDC     hdcPrn;
LONG    NumForms, NextForm;
BOOL    CurrentForm;

/* WinQueryProfileString is issued twice to obtain the names of the    */
/* default printer, the default presentation driver, and the queue     */
/* associated with the printer. If either of these requests fail, the  */
/* default values already defined in DEVOPENSTRUC are used.            */
/* We are only interested in the SPOOLER Output Printers               */

  if (WinQueryProfileString(habCurrent, (PSZ)"PM_SPOOLER", (PSZ)"PRINTER",
                           (PSZ)"", (PSZ)szKeyname, sizeof(szKeyname))){

    szKeyname[strlen(szKeyname)-1] = 0; /*Make last Semicolon into zero*/

    /* Get the name of the default printer                              */
    /* The form of each substring is:                                   */
    /* <port>;<dvr name[.???]>[,<dvr name>[.???]];<Q port>;<net parms>;*/

    if (WinQueryProfileString(habCurrent, (PSZ)"PM_SPOOLER_PRINTER",
                              (PSZ)szKeyname,
                              (PSZ)"", (PSZ)szDetails, sizeof(szDetails))){
      strcpy(szTempString, szDetails);

      /* Find the Default Driver Name                                   */
      pchScan = strchr(szDetails, ';');
      pchScan++;
      *(pchScan + strcspn(pchScan, ".,;")) = 0;
      strcpy(szPrinterDvrName, pchScan);
      strcpy(szDetails, szTempString); /* copy back original string  */
      if(SpoolerActive(habCurrent)){
        /* Find the Logical Address                                     */
        pchScan = strchr(szDetails, ';');
        pchScan++;
        pchScan = strchr(pchScan, ';');
        pchScan++;
        *(pchScan + strcspn(pchScan, ".,;")) = 0;
        strcpy(szLogPrinterAddr, pchScan);
        strcpy(szPrinterDataType, "PM_Q_STD");
        lPrinterDevType = OD_QUEUED;
      }
```

Figure 8. Example of Querying a Printer's Capability to Produce Raster Output (Continued)

```
      else{
        /* Find the Physical Address                                    */
        pchScan = strchr(szDetails, ';');
        *pchScan = 0;
        strcpy(szLogPrinterAddr, szDetails);
        strcpy(szPrinterDataType, "PM_Q_RAW");
        lPrinterDevType = OD_DIRECT;
      }
      /* An info request device context is defined.                   */
      hdcPrn = DevOpenDC(habCurrent, OD_INFO, (PSZ)"*", 6L,
                         (PDEVOPENDATA)&dopPrinter, (HDC)NULL);
      if (hdcPrn == (HDC)NULL){
        ReportPrinterError(1);
        return(FALSE);
      }
      /* See if default printer has changed since last check          */
      if(strcmp(szOldDetails, szTempString) != 0){
        strcpy(szOldDetails, szTempString);
        /* Retrieve key info from printer                             */
        DevQueryCaps(hdcPrn, CAPS_FAMILY, CAPS_VERTICAL_FONT_RES + 1, lDevCaps);
      }
      /* Retrieve key info from printer                               */
      DevQueryCaps(hdcPrn, CAPS_FAMILY, CAPS_VERTICAL_FONT_RES + 1, lDevCaps);

      /* See if printer can support raster output                     */
      switch((USHORT)lDevCaps[2]){
        case CAPS_TECH_RASTER_PRINTER:  /* RASTER PRINTER*/
        case CAPS_TECH_POSTSCRIPT:      /* RASTER CAPABLE?*/
          RasterOk = TRUE;
          break;
        default: /* All others */
          ReportPrinterError(2);
          RasterOk = FALSE;
          DevCloseDC(hdcPrn);
          return(FALSE);
          break;

      }
      else{ /* Could NOT find current default printer in spooler area  */
        ReportPrinterError(0);
        return(FALSE);
      }
    }
    else{ /* Could not find PM_SPOOLER Profile  */
      ReportPrinterError(0);
      return(FALSE);
    }
    return(TRUE);
}
```

Figure 8. Example of Querying a Printer's Capability to Produce Raster Output

This code fragment illustrates how to access the current default printer by opening a printer DC. Once the DC is open, the device can be queried to determine if it supports raster (GpiBitBlt) capabilities. It is also possible to obtain other information about the currently selected form after the DC is opened.

Glen R. Horton, Sr., *executive director, Inner Media, Inc., 60 Plain Road, Hollis, NH 03049. (603) 465-3216. Mr. Horton holds an MS in Computer Science from the University of Houston. He has been involved in the design and development of personal computer hardware and system level software for over ten years.*

Multimedia & Graphics

XGA Benefits OS/2 Applications

by Peter J. Hubbard

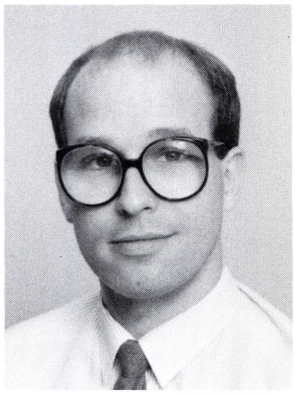
The new Extended Graphics Array (XGA), an evolution of the Video Graphics Array (VGA), gives the user enhanced resolution, color content, and hardware function. It is a high-performance, 32-bit bus-master video subsystem available on the motherboard of the recently announced PS/2 Model 90 XP486. It is also included as a standard adapter (the IBM PS/2 XGA Display Adapter/A) with the Model 95 XP486 family of machines, and as an option for 386SX, 386, and 486 Micro Channel PS/2 machines.

The XGA design has been optimized for use by window managers, such as OS/2 Presentation Manager (PM) and other Graphical User Interfaces (GUIs). XGA allows for application growth to more advanced function, while maintaining compatibility with existing VGA, EGA, and CGA software.

HARDWARE DEPENDENCIES

XGA's design is compatible with existing VGA software applications. It also offers many new advanced video features for PS/2s. These features allow for pop-up icons, pull-down menus, and other highly interactive components of today's applications.

XGA supports all VGA modes. The support of higher XGA resolutions and color content depends on the attached monitor and the amount of video memory installed. Figure 1 defines these combinations. Figure 2 shows a sample XGA screen.



Peter J. Hubbard

The XGA video subsystem is a 32-bit Micro Channel bus master.

Monitor	Description	Video Memory	Maximum Content
IBM 8503	12-inch mono	512 KB	640x480 64 gray shades
IBM 8507	19-inch mono	512 KB	1024x768 16 gray shades
		512 KB	640x480 64 gray shades
		1.0 MB	1024x768 64 gray shades
IBM 8512	14-inch color	512 KB	640x480 256 colors
		1.0 MB	640x480 65,536 colors
IBM 8513	12-inch color	512 MB	640x480 256 colors
		1.0 MB	640x480 65,536 colors
IBM 8514	16-inch color	512 KB	1024x768 16 colors
		512 KB	640x480 256 colors
		1.0 MB	1024x768 256 colors
		1.0 MB	640x480 65,536 colors
IBM 8515	14-inch color	512 KB	1024x768 16 colors
		512 KB	640x480 256 colors
		1.0 MB	1024x768 256 colors
		1.0 MB	640x480 65,536 colors
IBM 8604	15-inch mono	1.0 MB	1024x768 64 gray shades

Figure 1. Resolutions, Color Content, and Monitors Supported by XGA

The new XGA system units and the IBM PS/2 XGA Display Adapter/A option include device drivers.

XGA has the following device driver support:

- XGA Adapter Interface v1 for DOS 3.3 and DOS 4.0 (enables use of DOS applications written to exploit IBM PS/2 Display Adapter 8514/A)
- Microsoft Windows/286 v2.11
- Microsoft Windows 3.0
- OS/2 Standard Edition 1.2 and 1.3 (PM)
- OS/2 Extended Edition 1.2 and 1.3 (PM)
- AutoCAD Release 10

PERFORMANCE AND QUALITY IMPROVEMENTS

The XGA video subsystem is a 32-bit Micro Channel bus master. It can fetch and store data that an application requires for display purposes in main system memory without the assistance of the main processor. This frees the main processor to handle other system activity without significantly degrading the video performance.

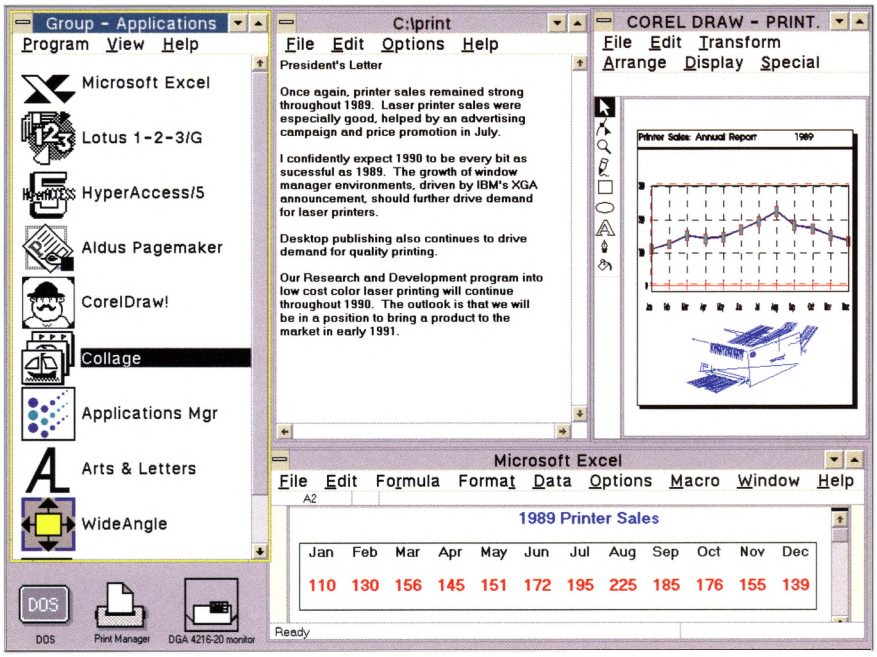

Group - Applications
Program View Help
Microsoft Excel
Lotus 1-2-3/G
HyperAccess/5
Aldus Pagemaker
CorelDraw!
Collage
Applications Mgr
Arts & Letters
WideAngle
DOS
Print Manager
DGA 4216-20 monitor
C:\print
File Edit Options Help
President's Letter
Once again, printer sales remained strong throughout 1989. Laser printer sales were especially good, helped by an advertising campaign and price promotion in July.
I confidently expect 1990 to be every bit as sucessful as 1989. The growth of window manager environments, driven by IBM's XGA announcement, should further drive demand for laser printers.
Desktop publishing also continues to drive demand for quality printing.
Our Research and Development program into low cost color laser printing will continue throughout 1990. The outlook is that we will be in a position to bring a product to the market in early 1991.
COREL DRAW - PRINT.
File Edit Transform
Arrange Display Special
Printer Sales: Annual Report 1989
Microsoft Excel
File Edit Formula Format Data Options Macro Window Help
1989 Printer Sales
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
110 130 156 145 151 172 195 225 185 176 155 139
Ready

Figure 2. Sample XGA Screen

The XGA has new high-resolution modes, more colors, a cursor sprite, and a hardware coprocessor for faster update of the video buffer. These features produce more natural images and higher screen content with faster response to the user.

COMPATIBILITY WITH VGA AND 8514/A

The XGA is compatible with VGA at the hardware register interface. Existing VGA applications should run unchanged. The XGA is compatible with DOS applications written to the 8514/A Adapter Interface (AI), although the hardware register implementation is different. The XGA does not support OS/2 protect-mode applications written to the 8514/A AI. XGA supports applications that take advantage of 8514/A under PM, but any applications written directly to the 8514/A hardware registers will not run on XGA. **Note**: IBM did not publish specifications for the 8514/A register interface.

Device drivers for XGA are available to support Microsoft Windows and OS/2 PM. Existing OS/2 PM and Windows applications written for VGA or 8514/A should run unchanged with these new drivers. The new XGA system units and the IBM PS/2 XGA Display Adapter/A option contain these drivers.

FOR MORE INFORMATION

The XGA register interface specifications will be published in the *PS/2 Hardware Interface Technical Reference: Common Interfaces* (IBM Order Number 84F9809) available in early 1991.

Peter J. Hubbard, *IBM UK Laboratories Ltd., Hursley, England. Mr. Hubbard is a product planner in the graphics adapter planning area. He joined IBM in 1985, initially working on the IBM Desktop Publishing Adapter and Image Adapter/A diagnostic code. In 1988 he joined the Presentation Products Merchandising team, where he was responsible for the merchandising strategy of several graphics adapter products. Mr. Hubbard holds a degree in Electronic Engineering from the University of Bath.*

Multimedia & Graphics

XGA Technology

by Nick Butler

The Extended Graphics Array (XGA), a new high-performance video subsystem, provides the high-performance graphics and high screen content (colors and resolution) demanded by Graphical User Interfaces (GUIs) such as OS/2 Presentation Manager (PM) and Microsoft Windows. It supports screen resolutions up to 1024x768 with 256 colors (depending on the amount of memory installed), including a new mode (using 16 bits per pixel) that allows 65,536 colors to be displayed at 640x480 resolution. XGA has a VGA compatibility mode as well as the extended graphics function. The 8514/A applications that use the 8514/A Adapter Interface will run on the extended graphics function via the XGA Adapter Interface. A new text mode that allows 132 columns of text to be displayed is also included.

XGA is the standard video subsystem on the new PS/2 Model 90 XP486 and PS/2 Model 95 XP486 families of machines. It is also available as a Micro Channel option card (IBM PS/2 XGA Display Adapter/A) for use in 386SX, 386, and 486 Micro Channel PS/2s.

SYSTEM DESCRIPTION

The following major components (shown in Figure 1) provide the extended graphics function:

- System bus interface
- CRT and memory controller
- Drawing coprocessor
- Video memory
- Serializer
- Palette
- Digital-to-Analog Converters (DACs)
- Sprite controller
- Sprite buffer

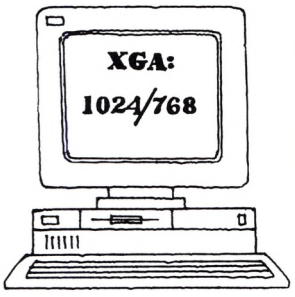
The *system bus interface* and the *CRT and memory controller* manage the XGA subsystem and the screen display. They provide the system processor with direct access to the video memory. All video memory is accessible by the system processor through one of three apertures: one for real mode, one for protect mode on a 16-bit processor or operating system, and one for protect mode on a 32-bit processor with a 32-bit operating system.

The *drawing coprocessor* provides a range of hardware drawing functions that operate on pixels in memory:

- Pixel Block Transfer (PxBlt)
- Line draw
- Area fill
- Logical and arithmetic pixel mixes
- Scissoring
- Map masking
- X,Y addressing
- Fast state save/restore

The XGA uses dual-port *video memory* (VRAM) to store pixel data. VRAM offers better performance than DRAM, since one port updates the displayed information while the other refreshes the screen. Depending on the country, the subsystem ships with 512 KB or 1 MB installed (it can be upgraded to 1 MB using the PS/2 Video Memory Expansion option).

Nick Butler



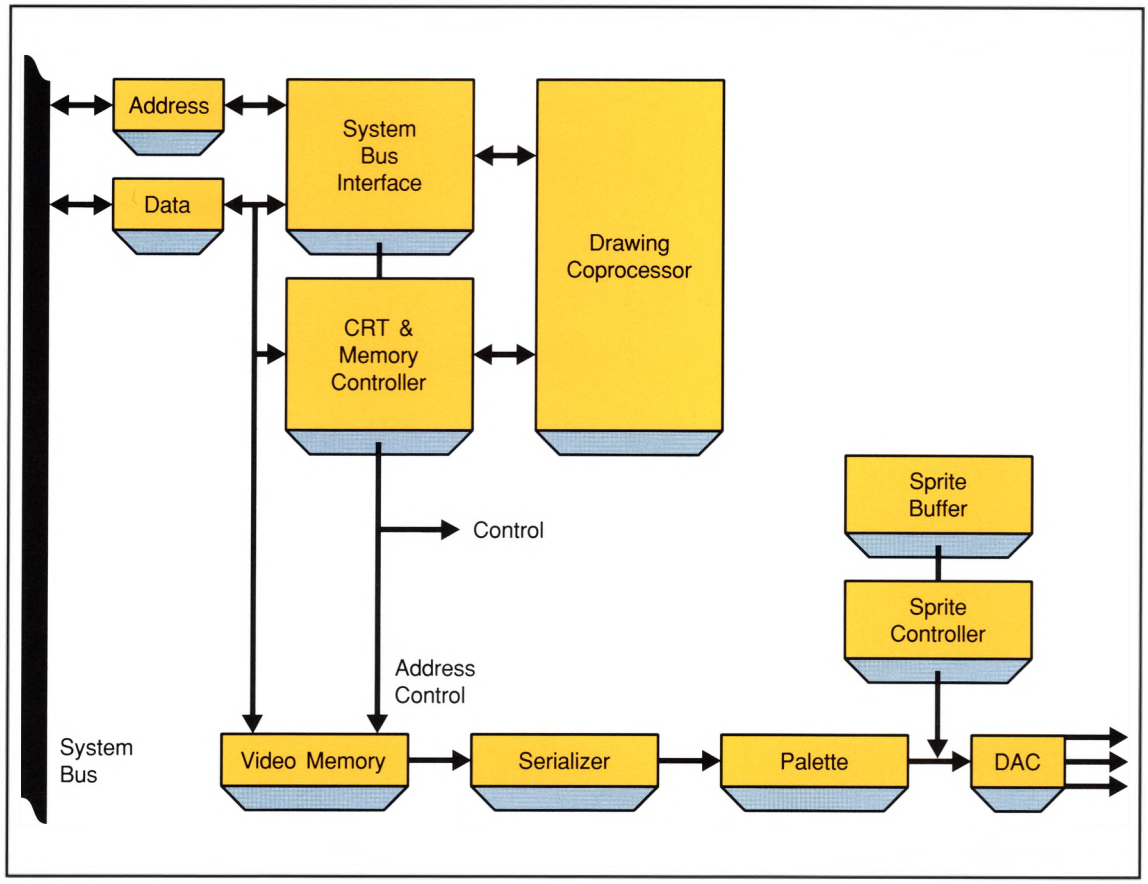


Figure 1. A Conceptual View of XGA

The XGA uses dual-port video memory (VRAM) to store pixel data.

The *serializer* takes pixel data from the serial port of the video memory, and passes it pixel-by-pixel to address the 256-entry *palette*. The color values from the palette are passed to the three 6-bit *DACs* (red, green, and blue), which produce the analog video signals for the monitor.

The *sprite controller* interfaces to the *sprite buffer* to display the 64x64 pixel sprite image on the screen.

DESIGNED FOR WINDOWING ENVIRONMENTS

XGA has many features for graphical user interfaces, allowing for highly interactive pop-up icons, pull-down menus, and other interactive components of today's applications.

- High-resolution screen modes allow more windows to be displayed on the screen at one time, and give greater text clarity.
- A hardware sprite allows a steady graphics cursor to be displayed without affecting the contents of video memory. This avoids the need for software collision detection.
- The powerful Pixel Block Transfer function works with three operands, and includes automatic "tiling" of the pattern and area fill based on a drawn outline.
- The pattern and source for line draw wraps automatically, allowing multi-colored patterned lines to be drawn easily.
- All drawing operations work on pixel data in areas of memory called *pixel maps*. The XGA uses bus-mastership on the Micro Channel to allow offscreen bitmaps to be held in system memory as well as display memory.
- The coprocessor is programmed using X,Y coordinates.

- A Save/Restore function allows the entire state of the coprocessor to be saved and restored quickly in multitasking environments.

Pixels, Pixel Maps, and X,Y Addressing

The drawing coprocessor works on pixels within pixel maps. A pixel map is an area of memory at a given address (the base address) with a defined width, height, and pixel format. Pixels can have 1, 2, 4, or 8 bits. The pixels can be ordered within bytes in one of two ways: left to right or vice versa. The XGA allows pixel maps of any arbitrary size up to 4096x4096 pixels.

The coprocessor is programmed using X,Y coordinates that are automatically converted into linear memory addresses (using the defined width and pixel size) before accessing the physical memory. See Figure 2.

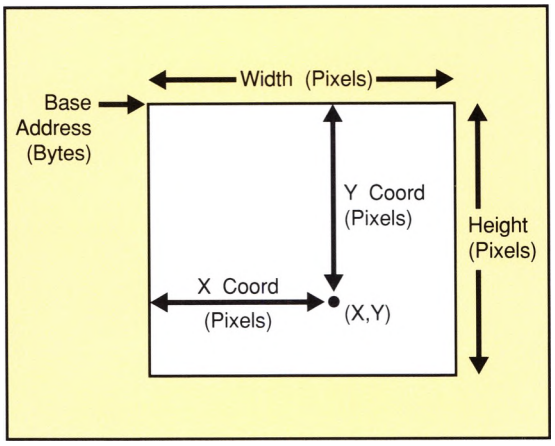


Figure 2. Pixels are Addressed Using X,Y Coordinates within Pixel Maps

The programmer can define up to four pixel maps at one time. Three maps (A, B, and C) are general purpose; the other is always used as the mask map. When starting a drawing operation, the programmer tells the coprocessor which maps are to be used as the source, the pattern, and the destination. In this way, map A, for example, could be the display pixel map. It could then be used as source, destination, or both, without having to move the pixel map parameters from one set of registers to another.

Pixel Maps in System Memory

A certain amount of space is available in the non-displayed (offscreen) areas of video memory. This is often used for storing fonts and offscreen pixel maps. GUIs, however, make extensive use of offscreen pixel maps (for example, for pull-down menus), and may be called upon to use fonts that are too large to fit in offscreen video memory. When the offscreen video memory is fully used, the GUI device driver may then start using normal system memory. With previous adapters, this has generally meant processing the pixels using the system processor. This is slow compared to special drawing hardware, especially if the operation is not a simple copy. It also ties up the processor and prevents it from preparing for the next drawing operation.

The XGA includes its own paging unit.

The XGA uses bus-mastership on the Micro Channel to overcome these problems. The drawing coprocessor's power can be used on pixel data anywhere in system or video memory.

To support paged memory environments, where the paging unit in the 80386 or 80486 is turned on, the XGA includes its own paging unit, using page tables of the same form. The XGA can operate using the main page tables used by the operating environment (with its cooperation), or on tables built by device drivers or applications themselves.

Pixel Block Transfer (PxBlt)

The PxBlt function works with three operands: the source, the pattern, and the destination. The source may come from a pixel map to copy data, or from color registers to set the destination to a particular color. The pattern may come from a pixel map, indirectly from the source, or it may be disabled.

For each pixel, the source and destination are combined using a Mix function selected by the 1-bit-per-pixel pattern. A 0 in the pattern selects the background mix, and a 1 selects the foreground mix. A full set of logical mixes is provided, supporting all the OS/2 and Windows Raster Operations (ROPs), with a selection of arithmetic mixes. Fast text drawing is crucial to windowing environments and other interactive applications. The pattern can select between foreground and background colors, allowing a 1-bit-per-pixel

text font in the pattern map to be rapidly expanded to colored characters in the destination.

The pattern and the source have another common feature. The X,Y addresses for these maps automatically wrap when they reach either side or top and bottom. This allows a small pattern to be "tiled" over a large area in the destination using a single operation (see Figure 3). The wrapping ability of the source X and Y allows multicolored "tiles" to be defined.

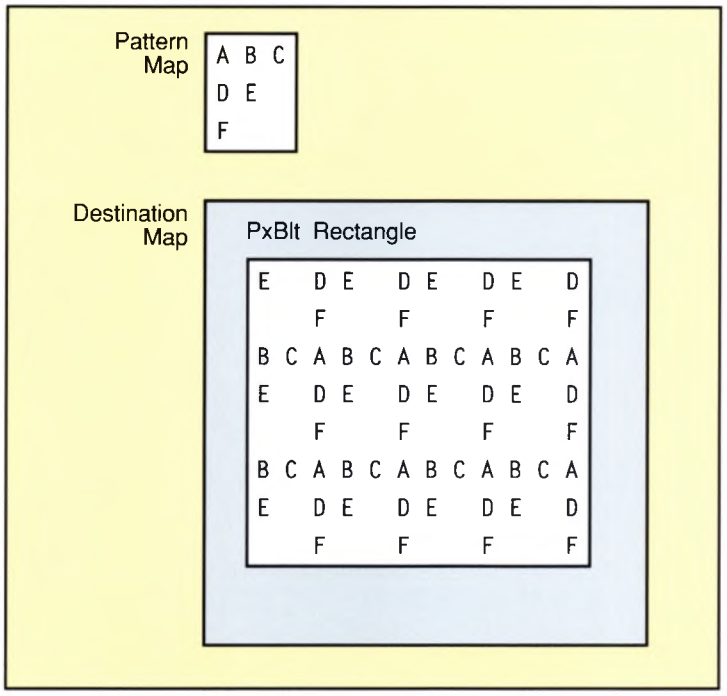

Pattern Map
A B C
D E
F
Destination Map
PxBlt Rectangle
E D E D E D E D
F F F F
B C A B C A B C A B C A
E D E D E D E D
F F F F
B C A B C A B C A B C A
E D E D E D E D
F F F F

Figure 3. Patterns are Automatically Tiled

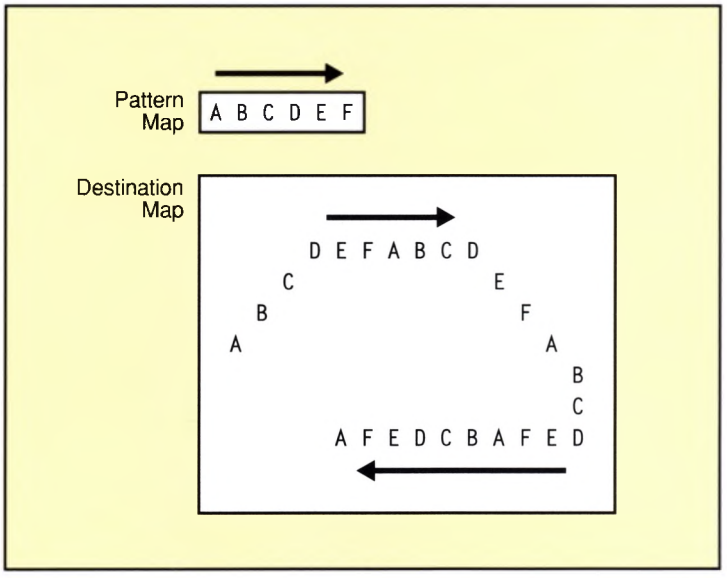


Figure 4. Line Patterns are Also Pixel Maps

Lines

The coprocessor draws lines using the Bresenham line-drawing algorithm. All pixels of a line can be drawn, or the first or last pixel can be suppressed to draw polylines correctly.

As with PxBlt, the pattern and source X,Y coordinates wrap at the edge of the maps. Line drawing, however, is different. While the X,Y coordinates of the destination move along the required path, the coordinates in the source and pattern move horizontally—left to right only. A simple 1-pixel-high pattern is drawn along the line, as shown in Figure 4.

Short lines can be drawn with simpler commands using "short stroke vectors," line segments up to 16 pixels long, and in one of eight directions (the X axis, the Y axis, and the 45-degree lines in between).

Area Fill

The XGA coprocessor allows arbitrary areas to be filled rapidly. The area is defined by an outline drawn in a 1-bit-per-pixel map. The outline is drawn using special versions of the line drawing functions (including short strokes).

When the outline is complete, a special PxBlt fills the inside of the defined figure. A parity fill algorithm is used, where a rectangle is scanned from left to right, starting from outside the figure, and alternately moving in and out of the figure as each successive boundary is crossed. The area can be filled with a pattern if required.

Scissoring and Map Masking

The coprocessor can, under programmer control, automatically "scissor" (that is, not draw) pixels that an operation attempts to draw outside a specified area. That area can be a simple rectangle, or a more complex shape defined in a pixel map (the mask map), with a pattern of 1s and 0s (1 allows the corresponding pixel in the destination to be changed; a 0 protects that pixel). This function can be used, for example, when drawing

into a background window that is partially obscured by other windows.

The Hardware Sprite

If there is no hardware sprite, software must maintain a graphics cursor in the display planes, removing it whenever drawing takes place to avoid overwriting it. When the cursor is moved (for example, in response to a mouse movement), the software must remove the existing cursor image, replace the original graphics data, and redraw the cursor image at the new location. This uses valuable processor time, reduces performance, and results in a cursor that flickers on and off and is unresponsive to mouse movements. The sprite in the XGA frees software from this burden and provides a steady, responsive cursor.

The hardware sprite can be any size up to 64x64 pixels. Its origin can be placed anywhere on the screen, including off the top and/or left sides. Each pixel in the sprite can be transparent, one of two programmed colors, or the 1s complement of the underlying pixel. The sprite colors are applied after the palette; each can be selected from a range of 262,144 colors.

State Save and Restore

In a multitasking environment, the system processor must change tasks from time to time. When changing tasks, the state of the current task must be saved so that it can be restored and continued at a later time. The state of the coprocessor can be rapidly saved and restored, making multitasking operation possible.

The coprocessor's operation can be suspended at any time and later resumed. Once suspended, the entire state of the coprocessor, including internal registers not visible in address space, can be saved to memory by reading repeatedly from a single I/O port. The 80X86 string input (INS) instruction can be used for this. The state is restored by writing the previously saved state data back into the same port (using the OUTS instruction). Once resumed, the operation will continue from exactly the point at which it was suspended.

ACCESSING THE XGA

XGA is controlled using a combination of I/O-mapped and memory-mapped registers. I/O-mapped registers are those that appear in the I/O address space of an 80X86 processor, and are accessed using IN, OUT, or other I/O instructions. Memory-mapped registers appear in the memory address space of an 80X86 processor, and are accessed using standard memory operations with all the available combinations of registers and addressing modes. In addition, many I/O-mapped registers are indexed (that is, the register is selected using an index in one I/O port, and the data for all indexed registers is written through a second I/O port). This technique, used also by the VGA, reduces the I/O address space required.

XGA is controlled using a combination of I/O-mapped and memory-mapped registers.

Memory-mapped registers are generally used to control the drawing coprocessor, where frequent access requires good performance. I/O-mapped registers (indexed and direct) are used for the remainder (mainly setup registers, where performance is less important).

Multiple XGA adapters (multiple instances) can be used in a system. Each instance has an instance number and has its registers mapped at different addresses. The memory-mapped registers are located at some point within the address range C0000 and DFFFF, the adapter ROM/RAM address space. The precise location is set by the PS/2 configuration process. Figures 5 and 6 illustrate how the I/O- and memory-mapped registers are located. When multiple XGA subsystems are installed in a system, the memory-mapped registers for all instances can be mapped within the same 8 KB block of address space. The allocation of addresses is the responsibility of the system configuration process, which ensures that there is no conflict between installed adapters (XGA or others).

Base address of the sixteen I/O registers of an XGA is 21x0, where x is the instance number. Figure 5 shows instance 6.

Base address of the memory-mapped registers of an XGA is (ROM Base addr) + 7K + (128 x instance number). Figure 6 shows instance 6.

Why Use Memory-mapped Registers?

Traditionally, display adapters such as the VGA and the 8514/A have been controlled through registers mapped into the 80X86 I/O address space. Memory-mapped registers have been introduced in the XGA for controlling the drawing coprocessor, where accesses are frequent and good performance is essential.

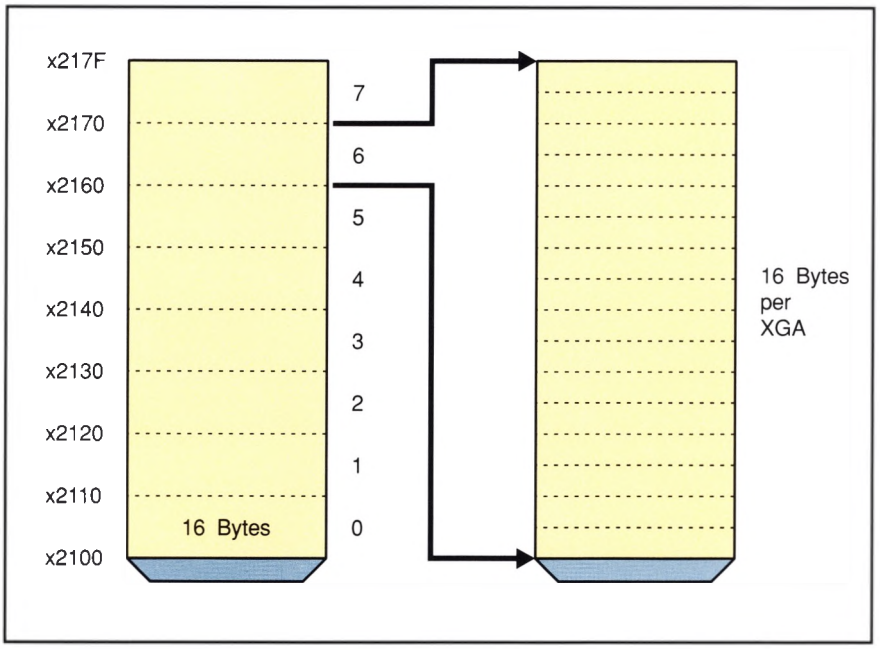


Figure 5. Addressing of I/O-mapped XGA Registers

The I/O address space in the 80X86 is limited to 64 KB, so individual adapters can only use a restricted number of addresses (to avoid possible conflicts between adapters). When the adapter contains many registers (typical of a display controller), indexed register addressing schemes are often used, as noted above. Memory address space is much larger (1 MB minimum in real mode), so adapters with memory-mapped registers can avoid using indexing, thus allowing direct access to all registers, and reducing code space and execution time.

When the 80386 is running in protect mode, the processor normally checks I/O accesses by applications to ensure they are allowed. It reads the I/O Permission Bitmap, a process that adds twenty cycles to each individual I/O access. Memory-mapped registers avoid this overhead, reducing to one-tenth the time taken to start many graphics operations.

Another advantage relates to multiple display adapters. Each instance of the XGA has a different set of register addresses, as shown earlier. Software must be able to run with any possible set. The 80X86 allows base-plus-offset addressing for memory accesses, using

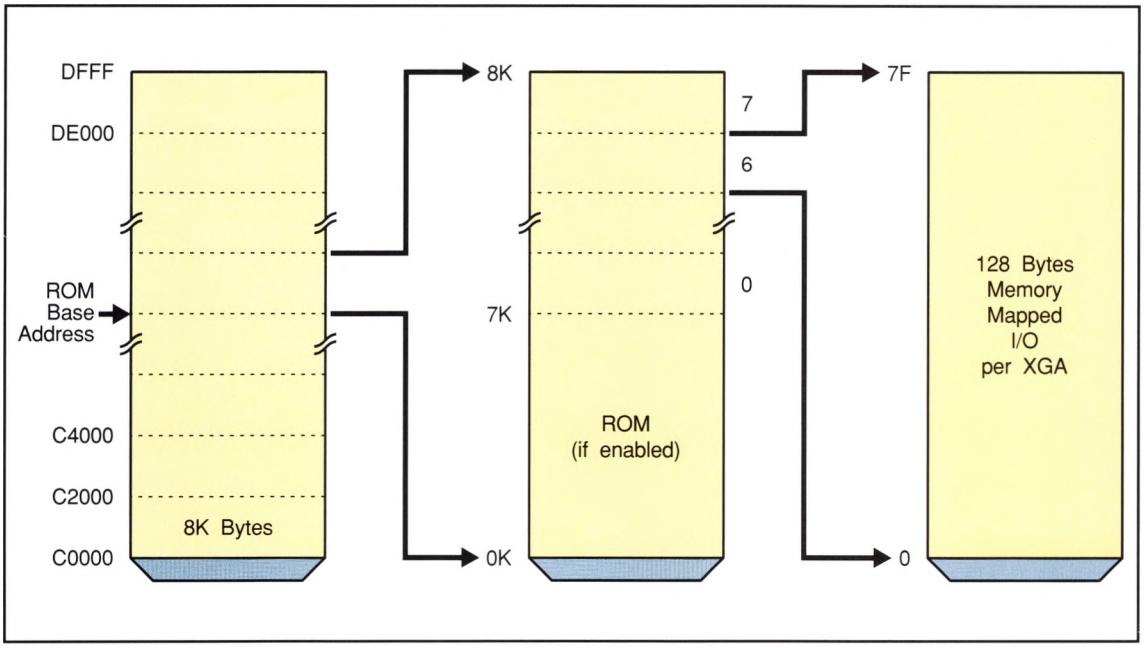


Figure 6. Addressing of Memory-mapped XGA Registers

a segment register and an immediate or register-based offset. The software would typically set the segment register to point to the first address of the memory-mapped registers, and then have immediate pointers to specific registers. I/O addresses, on the other hand, are always contained in the DX register; no form of base-plus-offset addressing is possible. For I/O-mapped registers, DX must be calculated correctly before each access, taking time and code space.

The register memory map for the XGA coprocessor, shown in Figure 7, illustrates how all the important registers are in plain view and easily accessible.

SCREEN RESOLUTIONS AND DISPLAYS

The XGA supports all VGA display modes. The higher resolutions available depend on the amount of video memory installed and on the modes supported by the attached monitor.

XGA introduces a new 16-bit-per-pixel mode allowing 65,536 colors to be displayed simultaneously on a 640x480 screen. This Direct Color mode does not use a color look-up table (palette). Of the 16 bits in each pixel, 5 are assigned to Red, 6 to Green, and 5 to Blue. This allows images from many sources (photographs, computer-generated materials) to be displayed with almost photo-realistic quality. The CRT controller supports this mode, but the coprocessor does not directly support it.

The memory required for the various screen modes and displays that XGA supports is shown in Figure 1 in the article "XGA Benefits OS/2 Applications" in this issue.

PROGRAMMING FOR THE XGA

The Video Subsystem section of the *PS/2 Hardware Interface Technical Reference: Common Interfaces* (IBM Order Number 84F9809) gives full details of the XGA register set. This includes useful advice for new application developers and those wishing to migrate existing applications. It should help to get the best from the XGA and to ensure compatibility with any future extensions. The *PS/2 XGA Display Adapter/A Technical Reference* (IBM Order Number 15F2219) gives information specific to the adapter card implementation of the XGA.

WHAT ABOUT THE FUTURE?

The architecture behind the XGA has been designed with future expansion in mind. Existing functions can be extended and new functions can be added so that applications written today will continue to run in the future if developers follow the advice given in the technical reference. Applications will only need to be modified if developers wish to take advantage of a new function.

132-COLUMN TEXT

The XGA allows 132 columns of alphanumeric text to display in one of the three standard VGA modes on a monitor that supports the higher video bandwidth required (42 MHz). Suitable displays are the IBM 8514, 8515, and 8507.

The 132 text columns allow much more data to be displayed in spreadsheet or word processor applications. They are used by 3270 Mainframe Interactive (MFI) terminal emulators.

The architecture behind the XGA has been designed with future expansion in mind.

VGA MODE

When running in its VGA mode, the XGA is compatible with the IBM VGA registers and BIOS calls, preserving the vast application base already available.

By using VRAMs and a wide 16- or 32-bit connection to the system bus, the XGA also boasts significantly increased performance for most VGA applications.

The XGA supplies VGA data to the Auxiliary Video Extension Connector (AVEC) on the Micro Channel, just as the VGA does, allowing option cards (such as the PS/2 Image Adapter/A) to use it.

The memory mapping within each mode is exactly the same as in the VGA, but it can vary between modes. BIOS Modes Set calls make allowances for the variations and should be used for all mode switches.

Byte 3	Byte 2	Byte 1	Byte 0	
Page Directory Base Address				0
Current Virtual Address				4
				8
		State B length	State A length	C
	Pixel Map Index	P1 Control		10
Pixel Map n Base Pointer				14
Pixel Map n Height		Pixel Map n Width		18
			Pixel Map n Format	1C
		Bresenham Error Term		20
		Bresenham K1		24
		Bresenham K2		28
Direction Steps				2C
				30
				34
				38
				3C
				40
				44
	Dest Color Comp. Cond.	Background Mix	Foreground Mix	48
Destination Color Compare Value				4C
Pixel Bit Mask				50
Carry Chain Mask				54
Foreground Color Register				58
Background Color Register				5C
Operation Dimension 2		Operation Dimension 1		60
				64
				68
Mask Map Origin Y Offset		Mask Map Origin X Offset		6C
Source Map Y Address		Source Map X Address		70
Pattern Map Y Address		Pattern Map X Address		74
Destination Map Y Address		Destination Map X Address		78
Pixel Operation				7C

Figure 7. Memory-mapped XGA Coprocessor Registers

Details covering the few other minor differences can be found in the Video Subsystem section of the *PS/2 Hardware Interface Technical Reference.*

THE ADAPTER INTERFACE AND 8514/A AI COMPATIBILITY

The XGA is compatible with the 8514/A through its Adapter Interface (AI). The XGA AI, supplied with the XGA, is used by applications operating in the DOS environment. The XGA AI also includes extra calls that take advantage of some new hardware functions.

Applications written to the 8514/A AI will operate with the following considerations:

- The XGA AI runs under DOS in real mode, not under OS/2 or Windows.
- The XGA is a single-display adapter. Applications requiring both the VGA and the 8514/A at the same time will still need two display adapters and two displays.
- Offscreen memory is used differently by the XGA AI. Applications using offscreen areas for their own purposes may not work correctly.
- If an application locates resources (such as fonts) in Expanded Memory Specification (EMS) memory and passes their addresses to the XGA AI, the application must use an EMS driver that provides the physical-address services at interrupt hex 4B.
- Certain XGA extended graphics functions run at different speeds than on the 8514/A. Applications that rely on specific timings will operate differently.
- The memory-resident code size occupied by the XGA AI is larger than that for the 8514/A adapter. This reduces the amount of system memory available to applications.
- The XGA AI is a .SYS driver, whereas the 8514/A AI is a Terminate-and-Stay-Resident (TSR) executable (.EXE) file.

XGA DEVELOPMENT

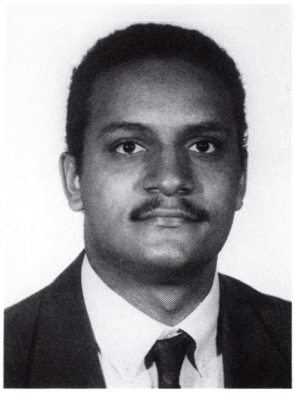
The XGA video subsystem chip set was designed and developed at the IBM United Kingdom Laboratories in Hursley, England together with the PS/2 XGA Display-Adapter/A.

Nick Butler, *IBM UK Laboratories Ltd., Hursley, England. Mr. Butler is an engineer in the Presentation Products department. He was lead engineer on the XGA chip design and development. He joined IBM in 1982. Mr. Butler holds a BA in Electrical Sciences from Cambridge University (UK).*

The XGA is compatible with the 8514/A through its Adapter Interface (AI).

Multimedia & Graphics

Programming the Audio Visual Connection



Danny Ellis

by Danny Ellis

The IBM OS/2 Audio Visual Connection (AVC) 1.02 provides multimedia authoring software, file management capability, and a runtime facility. This article describes application development using OS/2 Extended Edition (EE) and the AVC authoring language. Other components of AVC, such as those dealing with image capturing, image editing, audio recording, and audio editing are not discussed in this article.

AVC applications can communicate with users through hypertext links and digitized, high-quality audio and photo-like color images. A multitude of sound, display, and motion effects make this interchange eye-catching, fast paced, and attention holding. Developers can use the AVC to develop standalone applications or as a presentation service for other programs. The AVC, with the IBM Audio Capture and Playback Adapters and the IBM Video Capture Adapter/A, provides a total audio-visual solution for the application developer. The AVC runs under OS/2 1.2 or DOS 4.0 with expanded memory.

THE AUDIO VISUAL AUTHORING LANGUAGE

A *story* is the set of instructions that produces the desired sequence of images and sound, and controls the user's interaction with the system. These instructions are written in the Audio Visual Authoring (AVA) language using the AVC Story Editor. The AVA language is an interpreted, high-level language for developing audio-visual applications. The logic instructions and functions are a subset of IBM's REXX language. The audio-visual instructions deal with objects, such as image and sound. Like other interpretive languages, it requires a runtime module to execute the story. This runtime module is part of the AVC software package. It can be distributed (at no additional charge) with stories created with the AVC.

Novice Developers

AVC developers do not have to be programmers. The Audio-Visual instructions subset of the AVA language is sufficient for the novice user to easily develop exciting, eye-catching, and informative presentations. As shown in Figure 1, a story can be a simple list of images and sounds, prefaced by one of the following two instructions:

SHOW: Displays an image. Several parameters allow the user to control such things as how an image is dissolved on or off the screen, whether to display all or part of the image, and how long to show the image.

PLAY: Plays an audio selection. Parameters control such features as volume, and the start and end points, if the user chooses not to play the entire selection.

```
PLAY NUTCRACK
SHOW SOLDIERS
SHOW TOYS
SHOW DOLLS
PLAY MOZART1
```

Figure 1. Simple AVC Story to Play Audio and Display Images

Advanced Developers

A story can also contain complex logic. A story that contains lines of logic looks very much like a structured computer program instead of a simple list of images and audio. Some AVC stories, such as master display control stories and stories that prepare and file data used by other stories, may contain only AVA language logical statements. In this sense, AVC stories can be programs.

AVC also provides file management, string operations, variable data representation, and a host of other functions that make it suitable for advanced applications development. Additional functions can be added using AVA's interface to C and its user-written Dynamic Link Library (DLL) capabilities. This extends AVC's power to include all the functionality of OS/2.

Mainframe Connectivity

AVC can communicate with host applications. IBM's KnowledgeTool, a mainframe VM/MVS host-based expert system product, has the necessary code to communicate with the AVC Programmable Workstation (PWS) Connection feature. This allows AVC to be used as a presentation front-end to a KnowledgeTool application. Several AVA function calls are provided to support the Host-to-AVC PWS Connection services. Other mainframe host products can be written to use the AVC PWS Connection. This interface uses the Server Requester Programming Interface (SRPI) and can communicate with any mainframe program that supports the SRPI interface, further extending AVC's functionality.

AVC STORY EDITOR

The AVC Story Editor is an interactive, full-screen editor for coding AVA instructions into stories. Story Editor provides a graphical user interface and supports the use of a mouse or keyboard. The editor also shows the developer all the options and parameters required with AVA statements at the press of a key or the click of the mouse. It provides facilities for editing images and audio right from the story line. Since it is interpretive, the editor will flag any syntax errors when they occur with a message indicating the problem. These features of the Story Editor decrease the amount of knowledge and time required for a novice to code AVA stories. They do not hamper the experienced developer. For more seasoned programmers developing complex stories, Story Editor offers trace and debug facilities.

AUDIO-VISUAL CONTROL

Audio-Visual instructions are those AVA instructions that control displaying images and graphics and playing audio. As previously mentioned, the cornerstones of the Audio-Visual instructions are the SHOW and PLAY instructions. These instructions can be used in simple ways by novices (as shown in Figure 1), but also contain powerful parameters needed to meet the demands of complex stories.

AVC can communicate with host applications.

SHOW Instruction

The syntax for SHOW includes the following options:

imagename	Name of the image to display
method	Method of dissolve technique to use when displaying the image
dir	Direction for dissolve method
line	Line color to be used with dissolve
time	Time before the image is fully displayed
wait	Time to wait after image is fully displayed or audio sync label to wait on
size	Full or partial image display
imagexy	If size is partial, coordinates within image to display
wdth/hght	Width and height of partial display
screenxy	If size is partial, screen coordinates of the image when displayed

AVC has many dissolve methods for displaying images, including checker, explode, replace, fade, stripe, weave, multidirectional push, and others. Figure 2 shows different dissolve methods available with AVC. These methods used with the Time and Wait parameters allow for exact timing and sequencing of the displayed images. The Wait parameter also can specify an audio sync label. This label is a mark placed in the audio recording. It corresponds to a significant event, such as the beginning of a sentence or a change in music. By using sync labels added with the Audio Editor, displayed images can be kept in sync with the audio that is being played.

AVC has many dissolve methods.

Dissolve Method	Dissolve Direction
CHECKER	UP
CRUSH	DOWN
DIAGONAL	IN HORIZONTAL
EXPLODE	OUT HORIZONTAL
FADE	IN VERTICAL
FADEIN	OUT VERTICAL
FADEOUT	RIGHT
MOVE	LEFT
MOVEOFF	
MOVEON	
NONE	
PUSH	
REPLACE	
SPLIT	
STRIPES	
WEAVE	

Figure 2. Image Dissolve Methods Available with AVC 1.02

Using the size, imagexy, wdth/hght, and screenxy options, the SHOW instruction can display a full image or a portion of an image. Story Editor can automatically fill in the imagexy, wdth/hght, and screenxy coordinates for a partial display so that it is not necessary to calculate the coordinates. The image is displayed and the user chooses the partial area desired using a resizable/movable box. The user then can display the partial image on the screen using the mouse or directional arrow keys for exact positioning.

PLAY Instruction

The syntax for PLAY includes the following options:

audioname	Name of recorded audio file to play
method	Method to be used when playing the audio (FADE, FADEIN, FADEOUT)
channel	The audio channel to play (A or B)
volume	The volume level when played
time	Amount of time to fade the audio to the specified volume level
wait	Time to wait before proceeding to the next AVA statement
size	Play the entire recording or a portion of it (FULL, PART)
soundbeg	The starting point in the audio to be played (can be in seconds or a sync label)
soundend	The end point in the audio (can be in seconds or a sync label)

Like the SHOW instruction, PLAY has many options that allow for precise control over the audio volume, timing, and syncing. Using the fade, volume, and time options, audio can add excitement and pizazz to any presentation. For example, any music file can be gradually increased in volume to a crescendo, and then immediately dropped to an attention-grabbing transition into a sales presentation, speech, or product demonstration.

Other Instructions

Other AVA instructions are PASTE, CLEAR, COLOR, and WAIT. Again, these offer the user extended visual control over what is being displayed. PASTE is useful for overlaying images and text; it allows a user-defined color to be transparent when displayed over other images. Wherever this transparent color overlays an image, the underlying image shows through. CLEAR, COLOR, and WAIT are techniques for removing images from the screen, changing the background color over which images are displayed, and delaying execution of the story.

These instructions are the basic building blocks of any AVC story. Figure 3 is an example story using these audio-visual

instructions to present a sequence of images synced to audio.

These instructions are only a part of AVA's power—just as presentations are only a small part of AVC's usage. AVA incorporates instructions for logical structured programming that gives AVC the capability to be used in complex, advanced data processing applications. These include color image databases, interactive training systems, and applications that merge image and data for advanced decision-support systems.

STRUCTURED PROGRAMMING/ LOGICAL CONTROL

Structured programming requires variables. AVA variables called *symbols* can be up to 250 characters. Unlike C or COBOL, AVA variables do not have to be declared up-front; a variable can be declared within the story with an assignment statement. AVA variables can be simple or complex with high-level qualifiers forming compound variables. This is useful for working with arrays or lists. AVA can use variables to represent image names, audio file names, system control variables, data input by a user, and other uses where variable data is required.

AVA supports global variables called *common knowledge variables*. These variables follow the same naming conventions as local variables, but each has a prefix of an @ or # symbol and is accessible to all stories. AVA also has predefined system variables that will be discussed later. Figure 4 shows some typical AVA variable declarations. Since AVA is a subset of REXX, these variables follow the REXX rules for variable usage.

Numeric and Logical Operations

As shown in Figure 5, AVA has functions for integer and real number operations. AVA supports the Boolean operations of AND, Inclusive OR, and a host of comparative operators.

Execution Control

Any structured program needs execution control. The Call and Tell instructions transfer execution to another story or to a routine within the same story. CALL invokes a routine or story. Control normally returns to the next instruction after the original call. Of course, this is after the called routine completes. TELL is similar; control, however, is given to the routine or story being executed, and not returned to the calling routine or story. TELL links a series of stories. This is primarily used in an application that allows the user to browse randomly among a series of separate stories. CALL is normally used to call subroutines and common functions.

```
PLAY INTRO    {FADEIN,A,75,3,1.5,PART,BEGIN,TRSIT}
SHOW FACTORY  {WEAVE,OUT_H,NONE,1,A PRES,FULL}
SHOW PRES     {EXPLODE,IN_V,NONE,1,4,FULL}
SHOW BOARD    {REPLACE,RIGHT,BLUE,1,A PROD,PART,13.5,20.0,5.1,9.0,5.3,8.0
```

Note: The audio file INTRO contains the sync labels BEGIN, TRSIT, PRES, and PROD.

Figure 3. Sample AVC Story to Sync Images with Audio

```
PATIENT# = 1                          INTEGER ASSIGNMENT
TEMP = 98.6                           REAL ASSIGNMENT
NAME = 'BOB'                          CHARACTER  ASSIGNMENT
TEMP.1 = 98.6
TEMP.2 = 95.1                         ARRAY DEFINITION
TEMP.3 = 102.3
PATIENT.NAME.TEMP = 98.6
PATIENT.NAME.BF = '150/200'           COMPLEX ARRAY USAGE
PATIENT.NAME.EYES = 'CLEAR'
@COURSE, #PAGE                        GLOBAL VARIABLES
```

Figure 4. Typical AVC Variable Assignments

+, -, *, /	Addition, Subtraction, Multiplication, and Division
%	Division with integer portion of result returned
//	Division with real portion of result returned
Prefix -	Negate the following term
=	True if terms are equal (numerically or when padded)
==	True if terms are identical
/=	True if terms are NOT identical (inverse of ==)
>,<,<>,>=,<=	Standard Greater Than, Less Than, and Equal
&	Boolean AND
|	Boolean Inclusive OR

Figure 5. Operations Supported by AVA

AVC handles user interaction with trigger and input fields.

For example, in an interactive training application, a student selects a course from a menu of available courses. The individual courses are separate stories that are called using TELL from a master control story. CALL would be used within the individual course stories to call common subroutines for administering and grading tests or responding to specific user actions. These subroutine functions can be specific to the individual course story and coded within it. The function also can be common to all course stories and coded in a separate story.

Looping and conditional execution are available and implemented using the DO and IF instructions. DO allows for precise loop control with the DO WHILE and DO UNTIL versions evaluating an arithmetical or Boolean expression to determine the loop's termination point. There also is a DO FOREVER form that runs the loop until the Leave or Iterate instruction alters the execution. These instructions cause immediate or selective termination of the loop's execution. The If instruction evaluates comparative, arithmetical, or Boolean expressions for selective execution of instructions or groups of instructions.

If multiple conditions are to be tested, conditional execution also is available with the Select instruction. SELECT is the same as the CASE statement in C programming.

Other Functions

The AVA function calls provide string handling functions including parsing, concatenation, comparative operations, substring operations, positional operations, and others.

The availability of various file management functions, such as READ, WRITE, SEEK, and SIZE, provide a complete package of file handling capability. These file handling functions are meant for simple file/record manipulation, not for sophisticated file management. If these file handling routines do not suffice, the more accomplished C programmers may use their own routines. AVA allows for the addition of user functions, such as advanced file management. Figure 6 shows several useful AVA function calls.

USER INTERACTIVE CONTROL

Multimedia applications require user interaction that is different from the standard keystroke-oriented application. A multimedia application may ask the user to key the information, use the mouse to select the information, touch the desired selection on the screen, or even record an answer into a microphone. AVC handles user interaction with trigger and input fields.

A trigger field is a selectable area of an image that the user predefines and gives a unique name using the AVC Image Editor. Once the image is shown, the user can select the trigger field by placing the mouse cursor on the trigger and pressing the Enter key or left mouse button. The user knows that the area under the mouse cursor can be selected because the cursor changes shape. Trigger fields can also be used with a touch-screen. For this, the user simply touches the trigger, and the selection is the same as if a mouse made the selection. Once a trigger field is selected, the AVC variable @TF contains the name of the selected trigger field. This is the same name given to the trigger when it was created with

ARG()	Returns information about argument strings
BEEP()	Beeps the PS/2 speaker with specified frequency and duration
COMPARE()	Compares two strings
DATE()	Returns the date from the system clock
DELSTR()	Deletes specified substring within another string
FIND()	Searches the string for the first occurrence of a string
LASTPOS()	Returns the position of the last occurrence of one string in another string
LENGTH()	Returns string length
MAX()	Returns the largest number from the specified list
MIN()	Returns the smallest number from the specified list
OVERLAY()	Overlays a substring with another string
POS()	Returns the position of one string within another
SUBSTR()	Returns the substring beginning at the specified position and length
TIME()	Returns the time from the system clock in various formats

Figure 6. Additional Functions Provided by AVA

the Image Editor. Using @TF and the conditional If instruction, a story can recognize any selection made by a user with the mouse or touch screen. For example, the following instruction will pass control to the story GEARS when the student selects a trigger field named GEARS1.

```
IF @TF = 'GEARS1' THEN TELL GEARS
```

If the trigger field were named GEARS, then the following statement would accomplish the same action.

```
IF @TF<>"" THEN TELL VAR @TF
```

See Figure 7 for an example of the use of DO and IF in conjunction with trigger fields. Both assume the user enters the TEMP and @COURSE variables from the displayed image PROMPT and MENU respectively.

A user can also implement a selection with the keyboard using the @KEYCODE and @CHAR system variables. The variable @KEYCODE contains the keyboard code of the last keystroke entered by the user. It can recognize the F1–F12 keys or check for any character on the keyboard. The mouse down, up, click, and the touch-screen touch also provide @KEYCODE values that can be checked by AVA instructions (refer to the *AVA Language Reference Manual* (S15F-7134) for a complete list of keyboard codes). Figure 8 contains a sample AVA code fragment that checks for user-entered keystrokes using the @KEYCODE variable. The simple program in Figure 8 reacts to a user pressing the F3, E, or e keys to exit. The @CHAR variable has the same function as @KEYCODE, but @CHAR contains the actual character that the user last entered from the keyboard if it has an ASCII equivalent. For example, if the user entered "Y," @CHAR would contain "Y."

Input fields are areas defined on an image where the user enters data using the keyboard. The Image Editor can be used to create input field definitions and gives a specific name to each field. A variable by this name contains the data entered by the user and makes it accessible to the AVA story.

Input and trigger fields are used with the AVA instruction INPUT. The Input instruction tells AVA to activate any input or trigger

```
DO FOREVER
   SHOW MENU    {FADE,UP,RED,0,TRIGGER,FULL}  /* DISPLAY IMAGE WITH
   IF @TF =  'RADIO'  THEN CALL PRODUCT1       /* COURSE TRIGGER FIELDS
   IF @TF =  'TV'  THEN CALL PRODUCT2          /* IF ONE IS SELECTED THEN
   IF @TF =  'MCRWV'  THEN CALL PRODUCT3       /* CALL THAT COURSE MODULE
   IF @TF =  'VCR'  THEN CALL PRODUCT4         /* ELSE LEAVE THE LOOP IF
   IF @TF =  'EXIT' THEN LEAVE                 /* EXIT IS CHOSEN
END
```

Figure 7. DO and IF Use

```
MAIN LOOP:
@TIMEOUT = 1
DO FOREVER
   PLAY INTRO1  {FADEIN,A,75,3,1.5,PART,BEGIN,TRSIT}
   SHOW SCREEN1 {WEAVE,OUT_H,BLUE,1,1,FULL}
   SHOW SCREEN2 {EXPLODE,IN_V,BLUE,1,1,FULL}
   IF @KEYCODE = 101 THEN LEAVE                  /* CHECK FOR F3
   IF @CHAR = 'E' THEN LEAVE                     /* CHECK FOR 'E'
   IF @CHAR = 'e' THEN LEAVE                     /* CHECK FOR 'e'
   SHOW SCREEN3 {WEAVE,OUT_H,BLUE,0,1,FULL}      /* IF KEY NOT HIT LOOP
   SHOW SCREEN4 {EXPLODE,IN_V,1,1,FULL}          /* TO BEGINNING
END
DEMONSTRATION:
SHOW FOREST {STRIPES,OUT_H,RED,1,1,FULL}
SHOW TREES {PUSH,IN_V,RED,1,A BROOK,FULL}
```

Figure 8. Simple AVA Program Showing the Use of @KEYCODE and @CHAR Variables

Many multimedia applications can be written totally with the AVA language.

fields on the displayed image. AVA opens these fields for user input and/or looks for a user trigger selection. The user can give a specific field name to the Input instruction and only that field is opened for input. If no field is specified, all displayed fields (input *and* trigger) are opened for user input. An opened input field will have a cursor in it to show that it is available for input. Trigger and input fields can also be opened by specifying the keywords "trigger" or "input" on the Wait option of the SHOW statement. This yields the same results as the Input instruction, but saves lines of code. Note that "input" activates both input and trigger fields, whereas "trigger" only activates trigger fields. Figure 9 shows some sample AVA fragments that read input from the user and alter execution flow based on that input.

INTEGRATION TO EXTERNAL ENVIRONMENT

Many multimedia applications can be written totally with the AVA language. Applications such as presentations, small interactive training systems, and simple merchandise display systems are examples of applications that can be built with AVA code. More advanced multimedia applications like image databases, decision-support applications, and cooperative processing applications may require more functionality than AVA provides. Functions, such as database access (PS/2 and mainframe), spreadsheet access, sophisticated edit checking, complex file management, security, and standard micro-host communications capability, among others, are needed to enable development of

```
MAIN:
@L_NAME = ''              /* Employee last name
@F_NAME = ''              /* Employee first name
@MI = ''                  /* Middle Initial
@ADDRESS = ''             /* Employee Address
@EMPLOYEE_NUM = ''        /* Employee Number

DO FOREVER
        SHOW INQUIRE      /* Display image with input fields on it.
        INPUT             /* Put AVA into input mode.
        IF @L_NAME <> '' | @EMPLOYEE_NUM <> '' THEN TELL RET_DATA
           ELSE SHOW  ENTRY_ERROR
                          /* If data is entered then pass
                          /* control to retrieve routine
 END                      /* If no data, display error prompt
                          /* and refresh input screen.
```

Note: The image INQUIRE has the fields @L_NAME, @F_NAME, @MI, @ADDRESS, and @EMPLOYEE_NUM defined as input areas.

Figure 9. AVA Code to Read User Input Data and Alter Execution

these complex applications with high-quality image and sound.

The answer to this problem is AVA's integration with external functions. AVA can call external C programs and directly execute them. These functions can also be written as OS/2 DLLs and added to AVA using the AVA User DLL feature.

External C programs are executed directly by placing the name of the program in quotes on a line within the AVA story (for example, "C:\BIN\SQL.EXE"). AVA does not process the string inside the quotes, but passes it to the operating system where it is executed. Thus, the program SQL.EXE would be executed as if it were started from an OS/2 command prompt. Direct program execution is useful if the external C program controls the application and AVC is used only as a presentation vehicle for the images and sound. In this case, the external C program is written, compiled, and linked as normal and then run from the story.

AVA's interface to C programs is through a set of Application Programming Interface (API) calls. The external programs have access to all AVA variables (local and global) and can run other AVC stories. The VGET function call retrieves any AVA variable while VPUT updates any variable. CALLAVA runs an AVC story; ORETVAL sets a return value when the program terminates. The *AVA Reference Language Manual* and the file AVCOS2.OBJ within the product provide documentation for these calls.

Using the AVA User DLL feature is the second way to access external C programs. A User DLL can be used instead of direct program execution, if AVA will be used to control the applications and only some additional function is needed from the external program. A prime advantage of this feature is that it eliminates the operating system overhead required when calling a program. To add a User DLL function to AVA, the desired function is written as an OS/2 DLL and compiled and linked. Next, the DLL is defined to AVA using the FUNCADD statement. FUNCADD tells AVA the name of the function, the name of the DLL, and the entry point of the function code within the DLL. Figure 10 contains a sample AVA story that calls a user-written DLL. Notice that two functions, _TELL_ST and _VPUT_VAR, are within the same DLL and are defined with different FUNCADD statements. Also note that these functions are called from AVA by the names TELL_STORY and UPDATE_VAR, respectively. With the functions added to AVA, they then are called from within the

External C programs are executed directly.

```
variable = "@TIMER"
story1 = "time1"
load time keep

/* Define user-written DLL to AVA.  Must be done before the DLL is called
funcadd("TELL_STORY","C:\system.!av\userdll.dll","_TELL_ST");
funcadd("UPDATE_VAR","C:\system.!av\userdll.dll","_VPUT_VAR");

/* Execute the user DLL function
DO FOREVER
   number = time()
  UPDATE_VAR variable number      /* Call DLL to update variable @TIMER
  TELL_STORY story1               /* Call DLL to TELL the story TIME1
   if @keycode = 317 then leave   /* Check for F3 to exit
END
done:

/* Remove the DLL from AVA.  Call should be issued before the story ends
funcdrop("TELL_STORY")
funcdrop("UPDATE_VAR")
return

/*  User DLL story #2  TIME1    */
/*  This story displays the image with the correct time on it. */
/*  It is told by the DLL entry point _TELL_ST called from the */
/*  original story by the call TELL_ST                         */
show time
```

Figure 10. Sample AVA and C Code to Implement User-written DLL Functions (Continued)

Any AVC multimedia application now has all the function and capability of OS/2.

```
/*   This is a sample DLL called USERDLL written in IBM C/2.          */
/*   It is called by the AVC story and will either tell an AVC story */
/*   or update AVA variables.                                         */

#include <c:\ibmc2\include\mt\stdio.h>
#include <c:\ibmc2\include\mt\stdlib.h>
#include <c:\ibmc2\include\mt\string.h>
#include <c:\ibmc2\include\mt\math.h>
#define INCL_DOSPROCESS
#include <c:\toolkt12\c\include\os2def.h>
#include <c:\toolkt12\c\include\bsedos.h>

#define MAXSTRING 256
/*  VPUT_VAR   */
int far _loadds VPUT_VAR(int r_argc,char far *r_argv[]);
int far _loadds VPUT_VAR(int r_argc,char far *r_argv[])
{
 VPUT(r_argv[0],r_argv[1]); /** AVA call to update the variable r_argv[0]
                            /** with the data r_argv[1]
}

/*  TELL_ST   */
int far _loadds TELL_ST(int r_argc,char far *r_argv[]);
int far _loadds TELL_ST(int r_argc,char far *r_argv[])
{
 int ret;

ALLAVA(r_argv[0]);          ** AVA call to tell the story
r_argv[0] }

/*   SAMPLE MODULE DEFINITION FILE FOR USERDLL

LIBRARY USERDLL
DESCRIPTION 'USER WRITTEN DLL written in IBM C/2'
PROTMODE
IMPORTS
        APITRAN._VPUT
        APITRAN._CALLAVA
EXPORTS
        _VPUT_VAR
        _TELL_ST
DATA MULTIPLE
```

Figure 10. Sample AVA and C Code to Implement User-written DLL Functions

DO FOREVER LOOP. This example gets the system time and stores it in the variable NUMBER. Then, using a DLL, it updates the global variable @TIMER. Next, the story TIME1 is run from a DLL. TIME1 displays an image with the field @TIMER on it. This example displays the current time on the system clock and constantly updates it.

With this interface, any AVC multimedia application now has all the function and capability of OS/2. Interprocess communications (pipes, queues, and shared memory), Database Manager access, communications access to the host, printing capability, and other features of OS/2 are all available. Figure 11 shows block diagrams of a personnel database system that uses AVC and User DLLs to access employee data stored on the mainframe and employee pictures stored on the Local Area Network (LAN). AVC presents the user with screens asking for specific information that is used to formulate an SQL query. AVA reads the user's input, passes it to User DLL #1 that does edit checking on the data, and passes it to Session #2 using Named Pipes. Session #2 now communicates to the mainframe using Advanced Program-to-Program Communications (APPC) to retrieve the data from DB2, format it, and update the appropriate AVC variables with the information. Meanwhile, Session #1, using advanced file management functions, has found the employee's image on a local image server and passed the name to AVC for displaying.

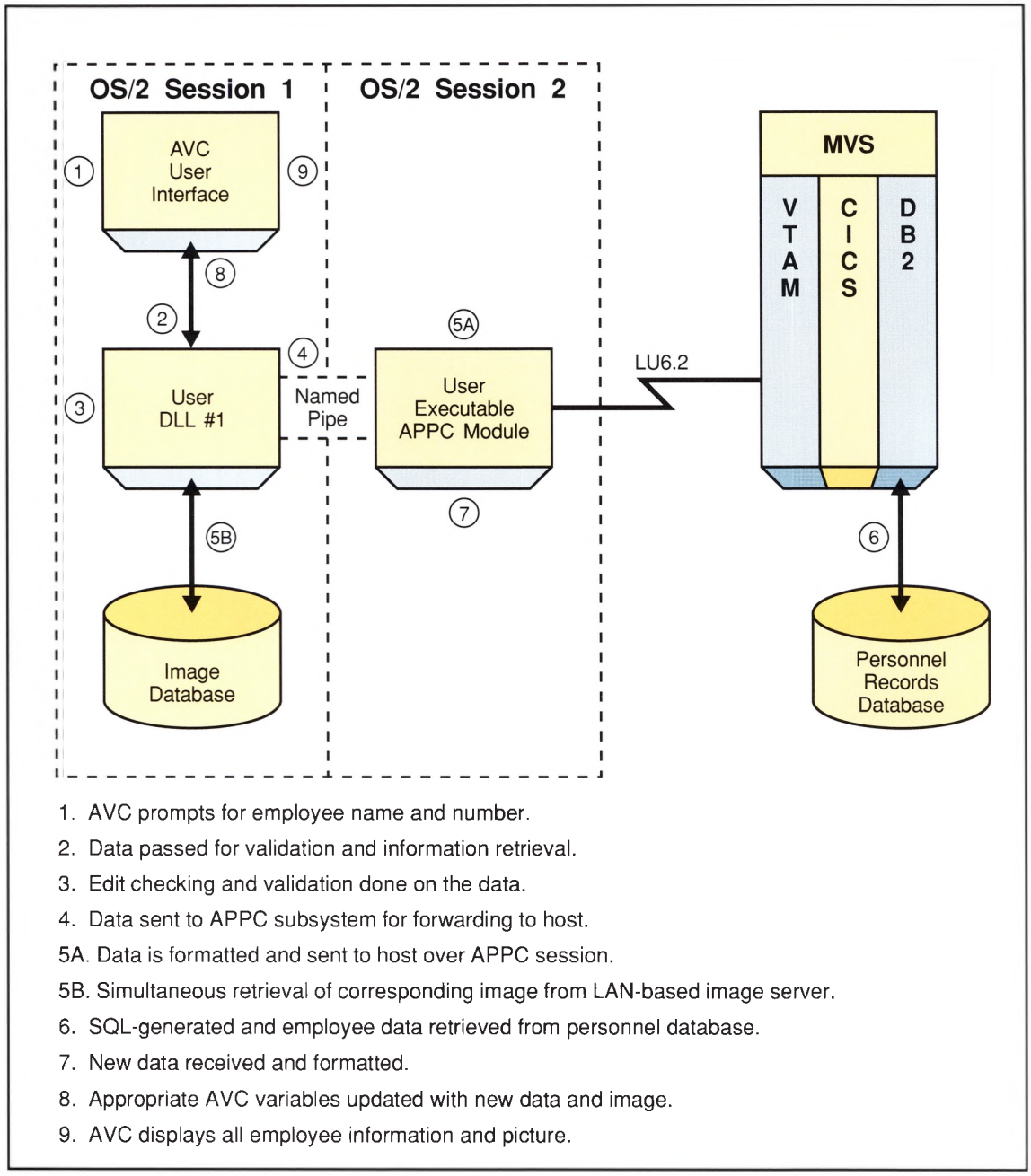

OS/2 Session 1
OS/2 Session 2
AVC User Interface
User DLL #1
Named Pipe
User Executable APPC Module
LU6.2
MVS
VTAM
CICS
DB2
Image Database
Personnel Records Database
1. AVC prompts for employee name and number.
2. Data passed for validation and information retrieval.
3. Edit checking and validation done on the data.
4. Data sent to APPC subsystem for forwarding to host.
5A. Data is formatted and sent to host over APPC session.
5B. Simultaneous retrieval of corresponding image from LAN-based image server.
6. SQL-generated and employee data retrieved from personnel database.
7. New data received and formatted.
8. Appropriate AVC variables updated with new data and image.
9. AVC displays all employee information and picture.

Figure 11. Example Application Using External Functions

ACKNOWLEDGEMENT

This article is based on material contained in the *AVA Language Reference Manual* and in the *OS/2 1.2 Programming Tools and Information*.

Danny Ellis, *IBM Corporation, One IBM Plaza, 5/1RU, Chicago, IL 60611. Mr. Ellis is an account systems engineer supporting one of IBM's largest customers. He has supported development of OS/2-based cooperative processing and multimedia applications for two and a half years. He has been with IBM for three years. He has a BS in Electrical Engineering from Washington University in St. Louis, Missouri.*

Software Tools

XIPC: A Comprehensive IPC Environment for OS/2

Ely Eshel

Daniel Schwartz

by Ely Eshel and Daniel Schwartz

XIPC™ is an advanced software development environment for applications employing Interprocess Communications (IPC) on a single system or across networks. It provides state-of-the-art IPC tools for the design, development, maintenance, and porting phases of IPC-intensive software development. XIPC reduces the time and cost of creating and maintaining sophisticated multitasking applications. By using XIPC, software developers can immediately enhance the quality and productivity of their IPC development efforts.

The major operating systems for mini- and microcomputers today are OS/2, UNIX® with its variants, and VMS. These operating systems provide a wide range of facilities for software developers to build sophisticated applications using concurrent processes, and to have these processes communicate with each other. Interprocess communication is the mechanism for passing information between processes.

The native IPC facilities provided by OS/2, UNIX, and VMS may not always be sufficient. Some features are available in one operating system but not in others, and some features are not available at all. In addition, the rules and programming practices (in which the available features are used) differ between the operating systems, making it difficult to develop software that can be used with all systems.

XIPC: A COMPREHENSIVE IPC ENVIRONMENT

The Extended Interprocess Communication Facilities (XIPC) gives the software developer a set of IPC tools, augmenting those provided by the three operating systems.

XIPC advances the state of IPC development tools in the following areas:

- **Software Engineering**: XIPC provides the systems developer with a set of IPC tools that is current with the latest programming methods. This includes online monitoring and interactive debugging at the IPC level. XIPC allows developers to view, in real-time, all their IPC resources as they are created and manipulated. It also moves all aspects of IPC configuration and tuning to the application level.

- **Functionality**: XIPC affords the developer a more complete set of IPC functionality. It includes new basic and advanced IPC operations and features.

- **Portability**: The Application Programming Interface (API) used to access XIPC is independent of the host operating system. Thus, the IPC components of a system developed using XIPC are immediately portable from one operating system to another.

- **Networking**: An XIPC application requires no program modification when it is distributed over a network. Distributed processes can communicate using XIPC functionality regardless of their network location or platform.

Using XIPC throughout all phases of the software development cycle results in several benefits:

- **Design**: XIPC's rich set of basic and advanced IPC functions simplifies system design.

- **Development**: The development phase is enhanced at the unit test level and during

system integration using XIPC's online monitoring and debugging facilities.

- **Maintenance**: The ability to monitor, diagnose, and support live systems in the field enhances system maintenance.

- **Porting**: Porting the IPC portion of an application between dissimilar operating systems is reduced to a recompile instead of a total redesign.

GENERAL CONCEPTS

XIPC consists of three independent IPC subsystems. Each subsystem includes a library of functions and support utilities. The subsystems are as follows:

- **QueSys (XIPC message queue system)**: A complete queue facility, incorporating many advanced features such as individual queue sizing, dynamic queue spooling, queue multiplexing, and so on.

- **SemSys (XIPC semaphore system)**: A comprehensive implementation of both event and resource semaphores, with a wide range of operations and various waiting and acquiring alternatives.

- **MemSys (XIPC shared memory system)**: A complete shared-memory management system, including memory allocation, access control, synchronization, locking, and protection down to the byte level.

XIPC Instances

An *instance*, an important XIPC concept, is an environment for performing XIPC activities. An XIPC instance has any combination of XIPC's three subsystems, according to the application's requirements.

XIPC instances typically are defined on an application-by-application basis, with each instance defined to meet the specific IPC requirements of its application. Thus, an XIPC instance is an IPC environment tailored to the specific needs of the programs that will use it.

In addition, special XIPC instances can be devised for testing various aspects of an application's performance or logic. Production copies of a system can be individually tailored for different customer and site requirements.

XIPC permits multiple instances to be started and to exist concurrently. Multiple active XIPC instances are completely insulated from one another.

Blocking Options

Many XIPC operations have the potential to block if they are unable to complete immediately. XIPC offers a complete set of synchronous and asynchronous options for handling these situations.

- **No Wait**: This option stipulates that if the operation cannot complete, the function should return an error code immediately.

- **Wait**: This option instructs the function to block the user process until the operation completes.

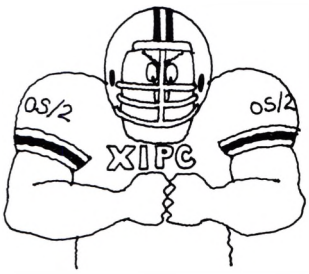
- **Time Out**: This option is similar to the Wait option, but allows the user to specify a maximum length of time to block. If the operation cannot complete within that time, the function should return an error code.

- **Asynchronous**: This option directs the function to place the operation in the background if it cannot complete. When it completes, an asynchronous signal is generated or an XIPC semaphore is set.

Online Monitoring

XIPC includes full-screen interactive monitors that provide continuous real-time views of the activities occurring within an XIPC instance. The respective monitors QueView, SemView, and MemView monitor an instance's subsystems.

The monitoring facility does not require special preparation of the applications being monitored. The facility can be invoked for any active XIPC instance, including those of production systems in the field, without any extra provisions, and without incurring performance overhead in the application when monitoring is not in use.



XIPC consists of three independent IPC subsystems.

XIPC monitoring becomes an invaluable tool for identifying IPC problems.

When invoked, monitoring becomes an invaluable tool for identifying problems, particularly when the problems result from incorrect or misunderstood usage of an application's IPC resources.

MONITOR FUNCTIONS AND LAYOUT

The three XIPC monitors are similar in layout and function. They share the same general "look and feel." Information appears in a matrix-like display where the processes ("users") involved and the IPC resources being manipulated form the axes of the matrix. Interaction between processes and IPC resources is displayed within the body of the matrix. Figure 1 depicts the general layout of an XIPC monitor screen.

Important subsystem capacity data displays at the lower right portion of the screen. Monitor status data appears at the top left of the screen. The command entry window is at the lower left of the screen.

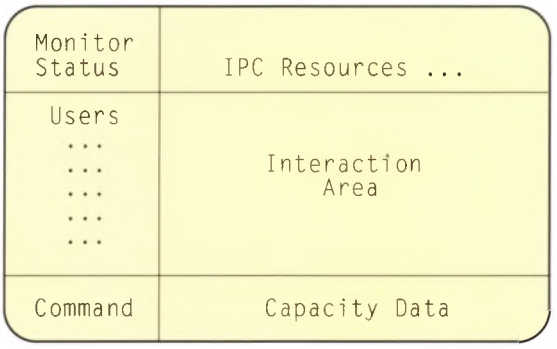


Figure 1. XIPC Monitor Screen Layout

MONITOR MODES

XIPC monitors update the display of XIPC activity in one of the following modes:

- **Interval Snapshot Mode**: Refreshes the monitor display at a user-specified millisecond interval.
- **Trace Flow Mode**: Refreshes the monitor after every XIPC operation in the monitored subsystem. The monitor pauses for a user-specified number of milliseconds, then it continues.
- **Trace Step Mode**: Completely stops activity in the monitored subsystem after each XIPC operation is performed. The user must press a key to proceed to the next XIPC operation.

Note that monitoring an instance's subsystem (in any mode) has no effect on the performance of other subsystems in that instance or on other concurrently active instances.

Zooming

XIPC monitors provide a set of facilities for examining aspects of an instance with additional scrutiny. One of these tools is the monitor zoom window. The zoom window allows the developer to watch general XIPC instance activity, while focusing on the activity of a specific aspect of the subsystem being viewed. Each monitor provides a wide array of zoom window options from which to choose. Figure 2 shows a QueView monitor screen with a zoom window opened for one of the queues in the instance.

Browsing

The browse facility provides a second, more powerful means of monitoring the status of an XIPC instance. Browsing is possible from within the QueView and MemView monitors for looking at the contents of message queues and shared memory segments. Figure 3 shows a QueView browse screen.

Browsing within QueView provides viewing access to complete message queues, individual messages, message text segments, and specific ASCII and Hex patterns occurring within a live XIPC QueSys instance. Using these capabilities, it is easy to confirm the format, length, contents, priority, and sequencing of messages on a queue while the messages are still on the queue. A user can search for text patterns (ASCII or Hex) locally within a message, global to an entire queue, by scanning forward or backward, chronologically, or by priority.

Similarly, browsing within MemView allows the developer to scan and search among shared memory segments. It also allows memory segment contents to be watched in real-time.

Breakpoints

In addition to the trace modes described above, programmers can specify logical IPC breakpoints, tied to a condition or an event involving one or more XIPC elements (such as users, semaphores, queues, or memory segments).

COMMANDS

Figure 4 lists some commands available for the three XIPC monitors.

THE MESSAGE QUEUE SYSTEM (QueSys)

A QueSys message queue is a set of messages organized by both arrival time and message priority. These are referred to as the time and priority strands of the queue.

Each QueSys message has two components: a message header that contains information about a message, including its priority, time of arrival, length of the message text, and a pointer to the text; and the actual message text, residing in the message text pool area.

Every QueSys message has a priority value associated with it, assigned when the message is placed onto a queue. Priorities are useful for selective message retrieval operations. These include specific priorities, priority ranges, and so on. The queue's time strand keeps messages on a queue in chronological order.

Queue Capacity

QueSys queues are created with individual capacity limits. The limits are specified as part of the queue-create operation in terms of maximum number of messages, maximum number of bytes, both, or neither.

Using this capability, it is possible to throttle message traffic within an application. Runaway message producing programs are controlled by setting a limit on the amount of traffic their outbound queues can hold. This also allows the programmer to protect an instance's message text pool from being monopolized by over-producing programs.

Overflow Spooling

Complementing the specific sizing capability is the ability to employ overflow message spooling on a queue-by-queue basis. When spooling for a queue is ON, the queue can absorb messages beyond its normal capacity by spooling to disk until space on the queue becomes available. At that time, the queue automatically absorbs the spooled messages.

```
Version: 1.1.0              *** XIPC QueSys Monitor ***        Jun 01 11:30:00
                          Instance: F:\SRC\XIPC\TEST
QUEUES ------>                 0          1          2          3
                  10       InQueue    OutQueue   Request    Response
USERS             Flow       9/25       14/25      10/25      16/25
00 QueView                 393/1024   561/1024   746/1024   859/1024

01 Remote1   WRT ...         ...        ...        ...        ...
02 Remote2   PUT  15         832        832        832        832
03 Local     ... ...         ...        ...        ...        ...
04 DataBase  ... ...         ...        ...        ...        ...
05 Services  ... ...         ...        ...        ...        ...

Msgs :  10/25   ( 40%) [■■■■■■■■■■.....]    Qid:   2
Bytes: 746/1024 ( 72%) [■■■■■■■■■■■■■■■.....]    Name:  Request
In:  588     Out: 578    Spool:  OFF       0 Msgs    CreateUid: 5
Wait List : 2                                      Created:   Jun 01 11:00:00

<Break> to enter Commands    Headers:   79/400  ( 19%)   Queues:  4/10  ( 40%)
                             TxtPool: 7616/8192 ( 92%)   Users :  6/20  ( 30%)
                             Fragmnt:   81/156  ( 51%)   Nodes : 15/100 ( 15%)
```

Figure 2. QueView Monitor Screen with a Zoom Window

```
Version: 1.1.0              *** XIPC QueSys Monitor ***        Jun 01 11:45:10

QUEUE  : Qid = 0     Messages = 13/25     Bytes = 915/1024     Name = InQueue

MESSAGE:  TIME  #9     Length = 254     Priority = 1000    Sequence# = 229

 00000:  4e657773  2d204465  632e2039  2c203139  38393a20  News: Dec. 9, 1989:
 00020:  20584349  20686173  20726570  6f727465  646c7920   XCI has reportedly
 00040:  66696c65  64207375  69742069  6e204665  64657261  filed suit in Federa
 00060:  6c20436f  75727420  61676169  6e737420  5a434f4d  l Court against ZCOM
 00080:  20666f72  20696e66  7269676d  656e7420  6f662062  for infringement of b
 00100:  7573696e  65737320  616e6420  66616c73  65206d61  usiness and false ma
 00120:  726b6574  696e6720  74616374  6963732e  20546865  rketing tactics. The
 00140:  20737569  7420636c  61696d73  20746861  74207468   suit claims that th
 00160:  65736520  70726163  74696365  73206861  76652072  ese practices have r
 00180:  6573756c  74656420  696e2075  6e737065  63696669  esulted in unspecifi
 00200:  65642064  616d6167  65732074  6f205843  49277320  ed damages to XCI's
 00220:  696e7465  726e6174  696f6e61  6c206d61  726b6574  international market
 00240:  696e672e  20323a33  3720504d  2e00              ing. 2:37 PM..

Command: g/News/                                              Offset = 0
```

Figure 3. QueView Browse Screen

XIPC QueSys provides much flexibility in dispatching and retrieving QueSys messages.

Common Commands
Set snapshot interval time
Enter "trace flow" mode
Enter "trace step" mode
Leave trace mode
Pan display to a user
Zoom on a user
QueSys Commands
Pan display to a queue
Zoom on a queue
Zoom on a queue's messages
Zoom on a queue's spool activity
Zoom on the message text pool
Browse messages on a queue
SemSys Commands
Pan display to a semaphore
Zoom on a semaphore
MemSys Commands
Pan display to a memory segment
Zoom on a memory segment
Zoom on the memory pool
Browse a memory segment
Watch a memory segment
Watch a memory segment's sections

Figure 4. XIPC Monitor Commands

Programs can start and stop overflow spooling dynamically in response to changing traffic loads.

Message Text Pool

The actual text of every QueSys message spends its entire existence in the instance's message text pool. Proper configuration of an instance's message text pool can make the difference between a well-performing system and one that performs poorly. There are two aspects to message text pool configuration: the size of the pool and the allocation unit used by the pool.

The message text pool size defines the total amount of memory allocated to the instance for holding the text of QueSys messages. The size of the allocation unit determines the multiple used to make all text pool allocations. It directly affects the level of fragmentation that occurs within the pool.

Queue Multiplexing

XIPC QueSys provides the developer with much flexibility in dispatching and retrieving QueSys messages. This distinguishes QueSys from other message queuing facilities. Of particular note is the ability to operate atomically on multiple message queues. For example, one can retrieve the highest priority message, or perhaps the oldest of the high priority messages, from a group of three queues, and so on.

Operations that move messages to and from queues are generalized to work with lists of queues. An operation involving only one queue uses a single element list.

THE SEMAPHORE SYSTEM (SemSys)

Two classes of semaphores are available using SemSys: event semaphores and resource semaphores.

Event Semaphores: These are Boolean in nature, and are used for signaling the occurrence of events. User programs can wait for clear event semaphores to be set by other user programs. The setting of an event semaphore usually will unblock (that is, wake up) at least one user program waiting for the event to occur.

Resource Semaphores: These are numerical devices for effecting resource access and control. Resource semaphores typically are used for limiting the concurrent usage of a resource to some preset level. User programs wishing to access or use the resource attempt to get a copy of the designated semaphore, perhaps blocking until a copy of the resource is available. User programs releasing held copies of the resource may, in turn, cause the wakeup of programs that were previously blocked when trying to acquire the resource.

XIPC SemSys supports operations involving multiple semaphores. Using this capability, it is possible to build sophisticated interprocess synchronization schemes. For example, it is easy to design systems that wait until ANY of five events have occurred, block until ALL of a group of resources are available simultaneously (atomically), block until ALL of a group of resources become available over time (cumulatively), or wait until ALL of a list of events have occurred.

THE SHARED MEMORY SYSTEM (MemSys)

XIPC shared memory activity revolves around MemSys memory segments. User programs read and write to and from MemSys memory segments as they see fit. To provide read-write access control at the byte level, XIPC MemSys introduces the concept of dynamic section overlays.

A segment is a repository of contiguous data. Reading and writing to areas of a segment require that the user program involved have the appropriate access to the entire area targeted by the read or write operation. Section overlays control accessibility to areas of a segment.

A section overlay is a logical window lying on top of all or part of a segment. It controls read-write access to the segment data beneath it. Each section that is defined on top of a segment has its own read-write privileges. These privileges control access to that part of the underlying segment that it covers.

A section overlay is defined at runtime. Its location, size, and read-write privileges are controlled at runtime. It is possible to modify privilege settings controlling access to underlying segment data as often as necessary. Finally, the section overlay can be removed from the segment when it is no longer needed.

Using sections, it is possible for a program to take control of all or parts of a given segment for as long a time as necessary. It may exclusively lock some areas, while imposing a read-only limitation on others. A program can lock portions of an in-memory database, individual records, or even specific fields for updating, as necessary. Tables or portions of tables can be made read-only or completely non-accessible for any time period required.

Segment Visibility Control

Using section overlays, it is possible to assign different levels of visibility to different portions of a segment's data. New sections can be defined to overlap existing sections. In this manner it is possible, for example, to have an entire table that is read-writable by all users, a single table entry that is read-only by all and read-writable by one specific user, and one field of this one record that is non-accessible by all, while still being read-writable by the one user—probably doing an update. Figure 5 depicts such a situation.

XIPC MemSys provides locking and read-write access control at the byte level.

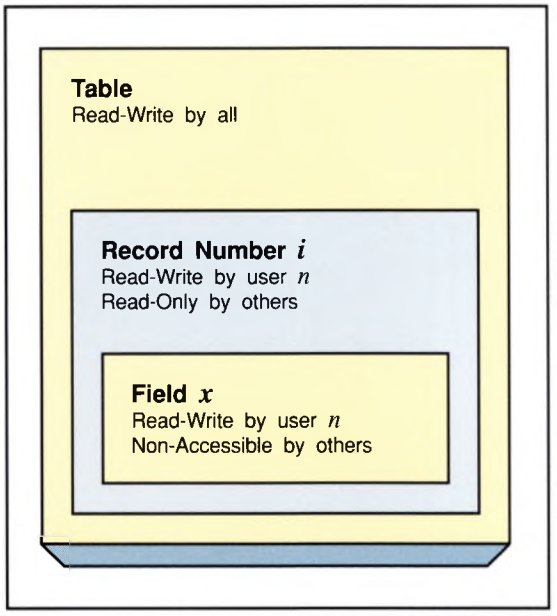


Figure 5. MemSys Segment with Section Overlays

Synchronized Read and Write Operations

MemSys read and write operations are guaranteed to be atomic in nature. User programs having read or write access to a targeted area of a segment are guaranteed that their read or write operations will execute atomically, regardless of the amount of data involved. MemSys enforces all serialization of segment access when necessary.

Distributed XIPC supports full XIPC functionality among diverse systems in networked environments.

Complementing the synchronization mechanisms, MemSys provides optional blocking for individual read and write operations. A caller program that is currently unable to complete a read or write operation because of an access barrier imposed by one or more overlying sections can opt to block until conditions permit the operation to complete. In this way, programs can be automatically synchronized when competing for access to areas of a segment.

Section Definition and Attributes

A section can have only one owner at any point in time. Ownership of a section is a prerequisite for modifying the section's privileges. Only users having read-write access to the section's entire area can acquire ownership.

Possible privilege settings are read-write, read-only, write-only, and not accessible. A user can attain exclusive control over a section by using these privilege settings.

Locking Segment Data

The most typical form of segment access control activity is for gaining exclusive access to a portion of a segment's data for a brief period of time, perhaps to work with that part of the segment in an atomic manner. This type of activity is called *locking*.

In most situations, locking and unlocking can be affected using a pair of high-level MemSys function calls. The functions automatically do the actual work of defining sections, becoming owner, and appropriately setting privileges.

Using MemView, it is possible to observe which parts of a segment are locked, who locked them, who is waiting for them, and their contents as they are manipulated.

DISTRIBUTED XIPC

The upcoming release of XIPC will provide XIPC functionality distributed across a network and allow heterogeneous systems to communicate across a network using the complete set of XIPC services. It is not necessary to modify existing XIPC applications to use distributed XIPC.

The major enhancement in distributed XIPC is the elimination of system boundaries regarding the location and platform of the participating processes.

Distributed XIPC supports full message queue functionality among diverse systems in networked environments, allows processes to manage resources and respond synchronously or asynchronously to events on different systems, and provides shared memory services among disparate platforms.

By using standard protocols as the underlying communications mechanism, many diverse systems can be transparently connected, using distributed XIPC, to work in concert without regard to their differences.

Ely Eshel, *Momentum Software Corporation, 401 S. Van Brunt Street, Englewood, NJ 07631. (800) 767-1462. Mr. Eshel is a founder and the president of Momentum Software Corporation. He specializes in software development and software engineering, and for the last few years has been focusing on OS/2 software, particularly multitasking tools and applications.*

Daniel Schwartz, *Momentum Software Corporation, 401 S. Van Brunt Street, Englewood, NJ 07631. (800) 767-1462. Mr. Schwartz, a founder of Momentum Software Corporation, is vice president of software development. He has an MS in Computer Science from New York University and specializes in multitasking software and interprocess communications.*

Software Tools

Front-ending Applications with EASEL

by Kevin Kornfeld

This article describes a step-by-step approach to implementing host application front-ends using EASEL®. The process described is based on work by the IBM Software Vendor Systems Center (SVSC) in Dallas on EASEL code development and cooperative processing. The concepts are part of a course for software vendors entitled "Cooperative Processing with EASEL" taught by the SVSC as part of the SAA Express program.

Imagine sitting before a terminal screen overflowing with columns of numbers, where the lack of color or structure makes the data almost incomprehensible. What if there were several such screens, each connected to another host—and your job was to take numbers from one screen and input them to the next for the entire day? All sorts of problems would be associated with such a scenario: lack of productivity, the waste of resources (time and talent), and a massive reliability issue.

Many dumb-terminal applications attached to mainframes work this way. They were the mainstay of most large computer environments. Today, personal computers (workstations) are making the undecipherable screen a relic of the past. The user interfaces possible with workstations provide ease of learning and use as well as high productivity. The use of front-end development tools, such as IBM's EASEL for OS/2 Extended Edition (EE), simplify the task of harnessing the power of Graphical User Interfaces (GUIs) for MIS shops having little experience with them.

OVERVIEW

EASEL allows developers to enhance user productivity and promote a "seamless" work environment. EASEL encourages and simplifies the implementation of GUIs compatible with IBM's Common User Access (CUA) guidelines, allowing a familiar user interface to be used across multiple applications. This reduces the learning curve associated with each.

In the normal course of business, many reasons will arise for changing a host application. Some of these are listed below:

- New requirements can arise (new features, functions, added or reduced product requirements)
- Enhanced productivity (time efficiency) desired
- Need for improved user interface (compatible with CUA guidelines) yielding reduced training time and less skill required to operate the system
- Graphical user interfaces desired (more efficiently implemented on a workstation)
- Movement of processing to a workstation (frees more host CPU time for other users)

The host application can be altered to support the desired changes, yet modifying proven software in a production environment entails risks. The most immediate risk is whether the effort can be accomplished in the forecasted time. Another risk is the availability of programmers intimately familiar with the original software. If not, time must be added to the schedule for learning. Since changing production code invariably introduces errors, there will be at least some reliability hit. Therefore, an organizational resource issue is introduced, not only because

Kevin Kornfeld

EASEL encourages and simplifies the implementation of GUIs compatible with IBM's CUA guidelines.

of the reliability question, but also for the downtime associated with the reinstallation, testing, and debugging of the modified host software.

Using EASEL to front-end the existing host application *without changing it* offers many advantages. The host software remains intact; and a new GUI is placed on the workstation. New functionality can be provided for the original application (adding value to it), while user productivity and ease-of-use are enhanced through the use of the workstation. Importantly, using the EASEL front-end allows users who do *not* have a workstation to continue using their non-programmable terminals as-is.

The ten-step procedure can help with the decision of when and how to use front-ends.

ANALYSIS OF EXISTING APPLICATIONS

The following ten-step procedure can help with the decision of when and how to use front-ends to existing host applications.

Step 1: Establish a Conceptual Model

Front-ending means replacing one existing input/output methodology with another. The host application must first be conceptualized in terms of input, output, and engine as shown in Figure 1.

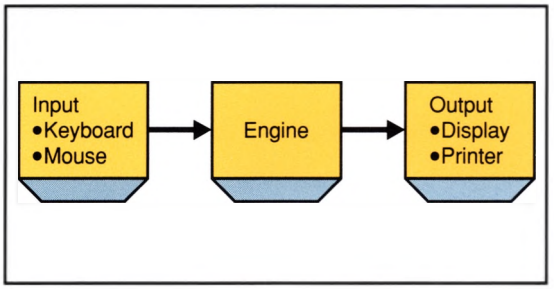


Figure 1. Conceptual Application Model in the Workstation Environment

The *engine* refers to the underlying algorithm, functionality, or purpose of the host application. It could be a database access, a series of computations, or another process type. The engine should be separated from the input methodology (the way information is obtained from the user) and the output methodology (the way results are presented to the user). By doing this, the decision on if and how to change the host application is based on the actual purpose of the application (see also Step 5).

Step 2: Evaluate the Host Application

Once a conceptual model has been established, it is possible to consider altering the application. The most fundamental consideration is whether the desired changes can be done in the workstation context without altering the host. As a general guideline, input or output changes can usually be implemented on the workstation, as can most additions to established functions. Fundamental changes to the engine itself could mean altering the host application, although such revisions could be minor.

Other considerations for a rewrite (at least partial) include:

- Availability of systems programmers (especially those knowledgeable in the application being evaluated)
- Maintainability considerations (could code changes be monitored, documented, updated, and reviewed more easily on the host than on the workstation)
- Distributed data access (should the local workstation database manager be used, at least on a temporary, working-set basis, for certain types of database access and retrieval)

Of course, the reasons for resisting any changes to a host application remain:

- Downtime on host code should be avoided.
- Bugs in host code are generally more serious (downtime and reliability) than bugs in workstation-based applications.
- Certain functions, such as display processing, are more efficiently and effectively implemented on a workstation.
- With less processing done on the host, more host CPU time is generally available for all users.

If any host changes are necessary, they normally would be performed (and thoroughly tested) before proceeding. Debugging both host and workstation software simultaneously should be avoided. Of course, certain host changes may not be testable without the support of the workstation application.

Step 3: Map the Host Screens

EASEL for OS/2 EE provides a FIELDS utility (see Figure 2) to help in analyzing host screens; that is, to break them apart into fields, and to determine the exact placement of text in rows and columns. EASEL uses this information to locate screen data; it then knows what interactions the host application process is expecting next. The FIELDS utility must be run for all host screens. The output can be printed for ease of reference.

Next, a screen control flow should be produced. This is a diagram illustrating all screens, and showing the order in which they are displayed—identifying which screen results from a given set of actions.

Step 4: Reduce the Screens

The screens of the screen control flow must now be analyzed. All screens must be identified and duplicates found. Similarities among various screens also should be noted. If two screens are doing a similar process, but displaying slightly different results, this would be considered relevant (see subroutine development under Step 7).

Step 5: Evaluate the Engine

It is important to understand clearly how and when the host application engine is being invoked. This is because part of the process of front-ending involves eliminating, combining, or reducing screens. The newly developed workstation front-end should fully exploit the optimum points at which the engine is to be invoked.

Part of this step is mapping and evaluating the host data flow. It is important to know how the front-end interprets user input and generates output, and at what stages in the

EASEL provides a FIELDS utility to help in analyzing host screens.

```
FIELDS Utility - Version H1.20  (HLLAPI)
Copyright (C) 1987,1988,1989 Interactive Images, Inc.
All Rights Reserved.

Session short name = A

000001111111111222222222233333333334444444444555555555566666666667777777778
567890123456789012345678901234567890123456789012345678901234567890123456789
================================================================================
                          PROFS MAIN MENU                                  A00
  Press one of the following PF keys.
  PF1  Process schedules                                Time:   9:36 AM
  PF2  Open the mail                Menu 1
  PF3  Search for documents                      1989    NOVEMBER     1989
  PF4  Process notes and messages               S   M   T   W   T   F   S
  PF5  Prepare documents                                    1   2   3   4
  PF6  Process documents from other sources      5   6   7   8   9  10  11
  PF7  Process the mail log                     12  13  14  15  16  17  18
  PF8  Check the outgoing mail                  19  20  21  22  23  24  25
                                                26  27  28  29  30
 PF10  Add an automatic reminder                              Day of Year: 311
 PF11  Look at Menu 2
       5664-309 (C) Copyright IBM Corp. 1983, 1987      PF9 Help    PF12 End
   * For Help Call (214)402-5219 or Enter PROBLEM on the Command Line


===>

================================================================================
Screen contains 51 fields.
Cursor is at [23,6].
```

Figure 2. Sample Output of FIELDS Utility (Continued)

```
Field summary:    Unp=Unprotected, Num=Numeric, Nds=Nondisplay, Mod=Modified

Fld#  Start     End       Len   Unp Num Nds Mod        First 30 Characters
-------------------------------------------------------------------------------
  1 | [1,2]     [1,76]     75 |   |   |   |   | [
  2 | [1,78]    [2,1]       4 |   |   |   |   | [A00 ]
  3 | [2,3]     [3,1]      79 |   |   |   |   | [Press one of the following PF
  4 | [3,3]     [3,6]       4 |   |   |   |   | [PF1 ]
  5 | [3,8]     [3,55]     48 |   |   |   |   | [Process schedules
  6 | [3,57]    [4,1]      25 |   |   |   |   | [Time:   9:36 AM          ]
  7 | [4,3]     [4,6]       4 |   |   |   |   | [PF2 ]
  8 | [4,8]     [4,34]     27 |   |   |   |   | [Open the mail              ]
  9 | [4,36]    [5,1]      46 | X |   |   |   | [Menu 1
 10 | [5,3]     [5,6]       4 |   |   |   |   | [PF3 ]
 11 | [5,8]     [5,48]     41 |   |   |   |   | [Search for documents
 12 | [5,50]    [6,1]      32 |   |   |   |   | [   1989   NOVEMBER    1989
 13 | [6,3]     [6,6]       4 |   |   |   |   | [PF4 ]
 14 | [6,8]     [6,48]     41 |   |   |   |   | [Process notes and messages
 15 | [6,50]    [7,1]      32 |   |   |   |   | [  S   M   T   W   T   F   S
 16 | [7,3]     [7,6]       4 |   |   |   |   | [PF5 ]
 17 | [7,8]     [7,48]     41 |   |   |   |   | [Prepare documents
 18 | [7,50]    [8,1]      32 |   |   |   |   | [              1   2   3   4
 19 | [8,3]     [8,6]       4 |   |   |   |   | [PF6 ]
 20 | [8,8]     [8,48]     41 |   |   |   |   | [Process documents from others
 21 | [8,50]    [8,58]      9 |   |   |   |   | [  5   6  ]
 22 | [8,60]    [8,60]      1 |   |   |   |   | [7]
 23 | [8,62]    [9,1]      20 |   |   |   |   | [  8   9  10  11     ]
 24 | [9,3]     [9,6]       4 |   |   |   |   | [PF7 ]
 25 | [9,8]     [9,48]     41 |   |   |   |   | [Process the mail log
 26 | [9,50]    [10,1]     32 |   |   |   |   | [ 12  13  14  15  16  17  18
 27 | [10,3]    [10,6]      4 |   |   |   |   | [PF8 ]
 28 | [10,8]    [10,48]    41 |   |   |   |   | [Check the outgoing mail
 29 | [10,50]   [10,80]    31 |   |   |   |   | [ 19  20  21  22  23  24  25
 30 | [11,2]    [11,48]    47 |   |   |   |   | [
 31 | [11,50]   [11,80]    31 |   |   |   |   | [ 26  27  28  29  30
 32 | [12,2]    [12,6]      5 |   |   |   |   | [PF10 ]
 33 | [12,8]    [12,48]    41 |   |   |   |   | [Add an automatic reminder
 34 | [12,50]   [12,80]    31 |   |   |   |   | [           Day of Year: 311
 35 | [13,2]    [13,6]      5 |   |   |   |   | [PF11 ]
 36 | [13,8]    [13,14]     7 |   |   |   |   | [Look at]
 37 | [13,16]   [14,6]     71 | X |   |   |   | [Menu 2
    | [14,8]    [14,50]    43 |   |   |   |   | [5664-309 (C) Copyright IBM Cor
 39 | [14,52]   [14,59]     8 |   |   |   |   | [     PF9]
 40 | [14,61]   [14,80]    20 |   |   |   |   | [Help    PF12 End    ]
    | [15,2]    [15,79]    78 |   |   |   |   | [  * For Help Call (214)402-521
 42 | [16,1]    [16,79]    79 |   |   |   |   | [
 43 | [17,1]    [17,79]    79 |   |   |   |   | [
 44 | [18,1]    [18,79]    79 |   |   |   |   | [
 45 | [19,1]    [19,79]    79 |   |   |   |   | [
 46 | [20,1]    [20,79]    79 |   |   |   |   | [
 47 | [21,1]    [21,79]    79 |   |   |   |   | [
 48 | [22,1]    [22,79]    79 |   |   |   |   | [
 49 | [23,1]    [23,4]      4 |   |   |   |   | [===>]
 50 | [23,6]    [24,45]   120 | X |   |   | X | [
 51 | [24,47]   [24,80]    34 |   |   |   |   | [
```

Figure 2. Sample Output of FIELDS Utility

process this I/O is occurring. The actual performance of the host application must be separated from the physical appearance of its screens; that is, a logical data model must be developed.

Step 6: Perform a CUA Analysis

The generation of CUA workstation screens to replace the old terminal screens is the next step in the front-ending process. It is necessary to organize and categorize the information contained in the screens already captured in Step 3. A judgement must be made on when and where to ask for information. The terminal screens can be translated to CUA one at a time. That, however, may not always the best approach. Can some screens be combined? Is there redundant information? Is certain information best presented in other formats? For example, a list of numerical codes might be presented more clearly as a list box of corresponding text descriptions indicating what the codes actually represent.

When information required for the host engine is gathered from the CUA screens, and the engine is then invoked, the information generated must also be transmitted to the user in CUA format. It is necessary to resolve which terminal screens to retain, combine, or revise. Note that the CUA Layout Tool, a feature of EASEL for OS/2 EE, allows the developer to generate and revise CUA-compliant screens without writing any actual code. This provides an ideal CUA prototyping environment.

Step 7: Design the Application

The application itself—the logic on the workstation side—can now be designed. This would initially consist of three main phases:

1. Connectivity (host to workstation, including session, logon, and logoff)

2. Main program (the basic program logic, in EASEL's event-driven "response to" format)

3. Subroutine structures (common code segments)

EASEL is a useful connectivity tool for providing host front-ends because it supports both 3270 and 5250 data streams as well as Advanced Program-to-Program Communications (APPC) for Logical Unit (LU) 6.2 applications. The wide availability of EASEL connectivity options enables multiple host applications to be front-ended using a single EASEL application. This capability could significantly reduce time and resource requirements by allowing a single point of interface for the user. Also, the task would generally be implemented entirely through EASEL.

Figure 3 illustrates some sample EASEL 3270 interface code that provides connectivity logic designed to enforce the asynchronous nature of Presentation Manager (PM) applications. All PM applications must respond to messages within 1/10 of a second. Since it takes far longer than this to wait for a host to connect, the developer must write code that allows the user to perform other tasks while waiting for the host to connect.

It is best to combine common segments of code into subroutines for ease of maintenance and program size considerations. Developers should give special attention to creating subroutines for certain host functions that occur on multiple screens. For example, Figure 4 shows a subroutine that builds a list box from certain supplied parameters for beginning and ending rows and columns. This subroutine can be used for any series of screens listing information in a tabular, predictable manner (such as the PROFS main menu).

Step 8: Refine the Application Design

With the initial design of Step 7, we can now introduce certain refinements to produce a more useful and technically appealing workstation application. For example, one such refinement is that EASEL version 1.1 can set up variables from the OS/2 command line. This feature can control which sections of EASEL code are called and when they are called. The EASEL "change to program" command (in combination with command line variables) can prevent certain sections of code from running when the EASEL front-end is invoked. This keeps such code out of system memory, possibly affecting performance and simplifying debugging.

The Layout Tool allows the developer to generate and revise CUA-compliant screens without writing any actual code.

```
copy true to ContOK                         # initialize Boolean

if (ContOK) then                            # if Boolean says to continue...
    action DefineWatch
    copy 1 to WatchRow
    copy 2 to WatchCol
    copy "V" to WatchChar
    action WatchForChar
    copy 25 to SettleTime
    action WatchForNoX
    copy 3000 to GiveUpTime
    copy "vm" to Keystrokes
    action EnterString
    add to Screen_3270
        insert "Entering VM Logon Screen...\n"
    action Watch                            # asynchronously wait for host
        begin resumable
            response to line "Watch done"  from A3270
                copy true to ContOK     # Boolean remains TRUE
                leave block
            response to line "Watch gave up" from A3270
                add to Screen_3270
                    insert "Host did not respond within 30 seconds.\n"
                copy false to ContOK    # Boolean goes FALSE!
                leave block
        end
end if
if (ContOK) then                            # only continue if Boolean TRUE
    action DefineWatch
    copy 24 to WatchRow
    copy 61 to WatchCol
    copy "C" to WatchChar
    action WatchForChar
    copy 25 to SettleTime
    action WatchForNoX
    copy 3000 to GiveUpTime
    action EnterString
    add to Screen_3270
        insert "Entering Screen for Logon...\n"
    action Watch                            # asynchronously wait for host
        begin resumable
            response to line "Watch done"  from A3270
                copy true to ContOK     # Boolean remains TRUE
                leave block
            response to line "Watch gave up" from A3270
                add to Screen_3270
                    insert "Unable to get to Logon Screen.\n"
                copy false to ContOK    # Boolean goes FALSE!
                leave block
        end
end if
```

Figure 3. Example of 3270 Host Communications

Since EASEL supports Dynamic Link Libraries (DLLs), other refinements could be the use of DLLs to enhance performance or to provide common code access for multiple processes. Functionality not attainable in the EASEL product also could provide another reason for using DLLs.

Developers who have reached this step in the front-ending process might want to consider possible value-added workstation features that could be included at the same time without requiring any host software changes. Examples of additions might be:

- Business graphics
- Cataloging data (possibly using the OS/2 Database Manager)
- Providing enhanced editing capabilities
- Use of a calendar or timestamp feature

```
### Subroutine for Reading Text from the Screen ###
                                        # TRow = top row
                                        # BRow = bottom row
                                        # LCol = left column
                                        # RCol = right column
subroutine ReadTextScr (integer : TRow, integer : BRow, integer : LCol,
        integer : RCol) is
    clear TRWork
    copy TRow to TopRow
    copy BRow to BottomRow
    copy LCol to LeftCol
    copy RCol to RightCol
    copy "TRWork" to TextRegionName
    action ReadScreen          # Read the screen into "TRWork"


### Subroutine for Building a List Box from the Host Screen ###
                                         # SRow = starting row
                                         # ERow = ending row
                                         # Spc  = spacing
                                         # LCol = left column
                                         # RCol = right column
subroutine BldLBScr (integer : SRow, integer : ERow, integer : Spc,
        integer : LCol, integer : RCol) is
    clear DB5LB1                          # Clear list box
    call ReadTextScr (SRow, ERow, LCol, RCol)    # Read the host screen
    copy (0-SRow) to Sumings[1]
    copy (ERow+1) to Sumings[2]
    call SUMINT(Sumings,SumEnt)
    copy SumEnt to MCount                  # Set up line count and max-count
    copy 1 to LCount
    while (LCount <= MCount) loop
        add to DB5LB1                                 # Build the list box
            insert textual line LCount from TRWork
        copy LCount to Sumings[1]
        copy Spc to Sumings[2]
        call SUMINT(Sumings,SumEnt)
        copy SumEnt to LCount      # Increment line count until max reached
    end loop


### DialogBox 5 is for Creating a List Box ###
                                # List box is an object within DialogBox 5
subroutine ProcessDialogBox5 (Boolean : DidOKCUA) is
    begin guarded
        response to start
            make DialogBox5 visible
            copy 3 to NumSRow
            copy 10 to NumERow
            copy 8 to NumSCol
            copy 43 to NumECol
            copy 1 to NumSpc

## Call the routine to build a list box

            call BldLBScr (NumSRow, NumERow, NumSpc, NumSCol, NumECol)
            select line 1 in DB5LB1

## other responses as required...

    end
```

Figure 4. Code for Building a List Box from Host Screen

Step 9: Implement the Application

After the workstation front-end has been designed and the CUA Layout Tool used to organize the user interface, the actual EASEL coding can be performed. This step requires using a text editor (such as the OS/2 System Editor provided with OS/2). Help should now be provided for user interface and understandability. Help already available on the host application can be incorporated into Help for the front-end, and enhanced, as necessary, to provide information specific to the personal workstation.

Step 10: Test and Debug

EASEL provides a rich set of debug options. An error log file captures error conditions and variables. All input and output can be recorded for later evaluation. An extensive trace facility records EASEL action statements, providing a step-by-step outline of program operation and logical processing as it occurs.

As with any other programming language, it is necessary to test the EASEL source code carefully, especially with regard to exercising all possible host application configurations and conditions. These tests should be compiled into test cases that are archived, and used when the host application or the workstation front-end is changed in any way.

ACKNOWLEDGEMENT

The author wishes to thank Eleonora Shafir and Mary Jackson for their contribution to this article.

Kevin Kornfeld, *IBM Corporation, U.S. Marketing & Services, 5 West Kirkwood Boulevard, Dallas, TX 76299. Mr. Kornfeld joined IBM in April 1989 as an advisory marketing support representative. A member of the Software Vendor Systems Center, he works with select software vendors in developing Systems Application Architecture applications for the OS/2 EE platform, providing design consultation, vendor education, and technical support. Mr. Kornfeld has a BSEE from the University of Arizona in Tucson and an MS in Computer Science from Southern Methodist University.*

Extended Edition Database Manager

Configuring Remote Data Services

by Marty Sirkin

This article is printed in two parts. This first part describes Remote Data Services (RDS), introduces vocabulary about RDS, shows how RDS works, and describes how to configure it. The second part (to be printed in the next issue of the Developer) describes how to customize a configuration for common environments, explains configuration for the DOS Database Requester, and describes some common pitfalls of configuration.

RDS is a new set of functions for OS/2 Extended Edition (EE) 1.2 that allows users to access databases not only on their own workstations (as in previous releases), but also on other workstations. Users can easily maintain access to a database even if the database is moved from one workstation to another. The Summer 1989 issue of the *Developer* contains a technical overview of RDS.

RDS frees users and application programs from knowing the actual location of the database. RDS stores database location in a database directory system maintained by a system administrator. This system is easily modified if a database is moved. This feature is known as *remote database location transparency*, so a user using Query Manager or a programmer writing a new application needs no knowledge of a database's location. They simply reference it by name. RDS does the rest.

In addition, RDS provides a powerful, remotely stored procedure interface that allows the developer to write programs that will operate on the database at the server. Developers can structure transactions so that the bulk of the database access work is performed on the server. This reduces the traffic on the network (fewer messages are passed between the user's workstation and the workstation on which the database is located), and results in faster execution times.

While location transparency makes writing an application easier, it does not free the developer from needing to know about RDS. To write and test an application that runs remotely, a developer must configure the Communications Manager to run RDS correctly.

With some knowledge about communications and the tools available with OS/2 EE 1.2, the installation and configuration of RDS can be simple and painless. This article does not cover the fine tuning that can be done to optimize performance with Communications Manager. Two invaluable aids in configuring the Communications Manager are the *IBM Operating System/2 Extended Edition Version 1.2 System Administrator's Guide for Communications* (S01F-0261), and the *IBM Operating System/2 Extended Edition Version 1.2 Database Administrator's Guide* (S01F-0267).

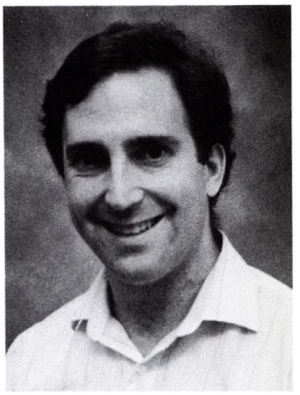
BASIC DEFINITIONS

The following terms refer to basic concepts that are needed to understand RDS.

Requester: A workstation that accesses remote databases. Requesters cannot create or access local databases.

Server: A workstation that stores databases for either local or remote access. With RDS, servers also can access remote databases.

Alias: This is a name used for an object other than its real name. RDS uses aliases for Logical Units (LUs), Partner Logical Units (PLUs), databases, transmission modes, node names, and so on. For example, both a local and a remote server would be likely to have a database called PAYROLL. Both databases cannot be cataloged with the same name because the Database Manager would not know which one to retrieve. Aliases solve this



Marty Sirkin

RDS frees users and application programs from knowing the actual location of the database.

problem by allowing references to the databases by whatever name the user prefers.

Profile: A set of variables in the Communications Manager configuration file that controls a particular feature. Common profiles used by RDS are the LU, PLU, transmission mode, and conversation security profiles.

Basic Configuration Services (BCS): BCS is part of the OS/2 EE installation that allows the user to configure RDS.

User Profile Management (UPM): UPM is an OS/2 utility that the Database Manager uses to restrict database access to qualified users. UPM uses a system of user IDs and passwords.

The installation and configuration of RDS can be simple and painless.

Transmission Mode: This is the language that the two workstations use to speak to each other over the network. RDS supports the following two types of transmission modes:

- **Advanced Program-to-Program Communications (APPC)**: The IBM Systems Network Architecture (SNA) strategic communications protocol.
- **SQLLOO (SQL LAN-only Option)**: This is a LAN-only enhancement used only by the Database Manager. It provides for faster data access and runs on the most popular Data Link Control (DLC) types: Token Ring, PC Network, and Ethernet. When Basic Configuration Services (BCS) is used to install an RDS system, that workstation will always be configured with SQLLOO. If APPC is desired on that workstation, it must be configured manually. SQLLOO provides almost all the crucial RDS functions available with APPC.

Logical Unit (LU): Logical units are the method that RDS uses to communicate. A logical unit is a software entity that is created by the Communications Manager. It is based on a profile that the user created in Communications Manager. To access a remote database, a logical unit must be on each machine. The communications line runs from one LU on the requester to another LU on the server (there may be more than one LU on a machine). Since OS/2 is a multitasking operating system, LUs can simultaneously handle more than one RDS database request to the same or different machines.

Partner Logical Unit (PLU): PLU contains all the information needed to establish communications with another machine. Among other things, a PLU record contains the address on the network of the other machine, the alias of the LU on the local machine, and the name of the LU on the remote machine.

Just as in real life, it is important to be able to listen as well as to speak in a conversation. Each requester must therefore have a PLU (so that it knows where the server is); each server must also have one (so it can locate and send messages back to the requester). A server that can be accessed by any one of 100 different requesters would need 100 different PLU records. That is a lot of profiles. RDS provides a shortcut (the next item).

Implicit PLU: An implicit PLU (used only on server workstations) allows communications to be received from any requester workstation. Instead of specifying the information about each requester (as would normally be necessary), the implicit PLU records that information when the requester first connects with the server. Therefore, the server knows where (on the network) and who (which LU) the requester is. Implicit PLUs have names that start with an asterisk (BCS sets up one called *PLU), and there can be an implicit PLU for each LU on the server. The implicit PLU set up by BCS has a transmission mode of *SQLLOO. The asterisk means that it is an implicit transmission mode (that is, the PLU will accept communications from any transmission mode, not only SQLLOO). The *SQLLOO transmission mode will work with any transmission mode, not just SQLLOO, so APPC transmission can be used on the requester.

Cataloging: The process of informing the Database Manager of the location of a remote database. Cataloging can be done either in the Query Manager or via an Application Programming Interface (API) call in an application program. A node catalog entry must be made for each remote server to be accessed. A database catalog entry must be made for each database to be accessed.

Data Link Control (DLC): DLC is a fancy name for the type of network being used to connect the machines. Examples of DLCs supported by RDS are Token Ring, PC Network, Ethernet, X.25, and SDLC. With certain restrictions, there may be up to two DLCs in a workstation at a time. They are

called adapter 0 (primary) and adapter 1 (secondary). Either DLC card can be primary, as set with switches (AT DLCs) or with the PS/2 reference diskette. Note that the adapter designation (0/1) is *not* affected by the physical slot in which the DLC card is located.

Session: This is a connection between two LUs on different workstations. There may be many sessions between LUs at once. For example, two Query Manager sessions on a requester can access the same database on a server. These sessions run between the same pair of LUs--one on the requester and one on the server. RDS ensures that a session cannot become active until both workstations have proper profiles to set up a valid session.

Conversation: Data being transferred between two programs over a session is called a *conversation.* Only one conversation can occur per session. RDS provides conversation security. A user (on a requester) must log on with a valid user ID and password for the server. Allowable user IDs and passwords can be specified in the server Communications Manager. Database Manager users should always use a special feature called *implicit conversation security*. An asterisk is entered as a user ID, meaning that the normal UPM database of server user IDs and passwords will be used to validate remote access. This eliminates the need to enter the user ID and password in both UPM and Communications Manager.

Both explicit and implicit conversation security can be used simultaneously. User IDs and passwords can be entered into the Communications Manager, and the asterisk can also be used. When a requester tries to access the server, RDS first tries to match an explicit user ID and password. If there is no match, it looks for an asterisk. If it finds one, it tries to match the logon against the UPM database. If these steps fail, it denies the requester access to the server.

Note: If explicit conversation security is used (by placing user IDs and passwords in the Communications Manager profile), users must also enter them into the UPM system on the server. If not, RDS will make the connection from requester to server, but the Database Manager cannot look up the user in UPM on the server. Therefore, it will fail and the database requester receives an error code. This is because the conversation security is used only to make a communications connection between the requester and the server. Once that connection has been made, the Database Manager still must access UPM to perform database operations.

Link Station: The DLC card controls the actual connection to another workstation. When a requester connects with a server, each workstation uses a link station to control the physical connection. Programs on the requester can query databases on the server, but all sessions go over the same link. The one exception is using different transmission modes (APPC, SQLLOO). Then, two links are needed (and two link stations on each machine): one for the APPC sessions and one for the SQLLOO sessions.

Link stations are declared in three places in the Communications Manager profiles:

- In the IEEE 802.2 profile within the LAN profiles (the number of link stations available)
- In the NetBIOS profile within the LAN profiles (the number of link stations for use by NetBIOS for both DOS database requester and LAN requester/server)
- In the DLC profile within the SNA profiles (the number of APPC link stations available)

To calculate how many link stations are available for SQLLOOs, subtract NetBIOS and APPC link stations from the 802.2 total, and multiply by 80%. If that number is zero or smaller, SQLLOO will not function.

Addresses: This is a twelve-digit (hexadecimal) description of the location of the machine on the network. RDS uses addresses to identify workstations on the network. Addresses on the network must be unique.

If the workstation is a database requester (or requester/server), the address of each server must be known to access databases.

Two types of addresses can be used:

- **Universal addresses**: An address that is specified on the DLC card itself (in ROM). IBM and other manufacturers have a numbering convention to ensure that the number on the card is unique. There are several ways to find this number. On some DLC cards, the number is physically

engraved on the card. Another way is to use a program on a diskette that often comes with the DLC card that will read and report the address. Or the system can be installed and rebooted, thus creating a file called \CMLIB\ACSLAN.LOG (for Token Ring and PC LAN) that lists the DLC address. For ETHERAND cards, the file is not created until an application uses the card. Then the real address is actually the address shown in the ACSLAN.LOG file with the bits in each byte reversed for ETHERAND.

- **Local addresses**: A network administrator chooses the address for each workstation, being very careful to avoid duplicates. Local addresses must start with the digit 4. Local addresses are convenient when equipment needs to be switched. Suppose that a DLC card in a server malfunctions. With local addresses, it can be replaced without affecting the requesters. If universal addresses were used, the PLU address field in each requester would have to be changed to the universal address of the new DLC card.

Workstation Name: A name selected during installation, used by Database Manager only for the DOS database requester. NetBIOS workstation names are also used by the LAN requester/server, but they are stored separately. It is best to use the same workstation name if both the DOS database requester and LAN requester/server are used.

Node Name: A user-selected name for the remote workstations. It is used in the catalog procedure. There are some restrictions on node names. They should be alphanumeric, including characters # @ $ _, and eight characters or less in length (with the first character of the name being alphabetic or one of the following: # @ $). Also, node names and workstation names are not required to be related in any way, although it makes them easier to remember.

The Query Manager refers to node names as workstation names. Remember that cataloging and uncataloging remote workstations in the Query Manager does not affect (and is not related to) the workstation name set during installation.

RDS starts a program on the server called the application agent to handle the RDS data requests, and to open the database.

USING RDS

Assume that there are two workstations: a requester running a front-end query application, and a remote server holding the database to be queried. Figure 1 shows the steps needed to establish the remote connection.

When the application program tries to open the remote database, it first looks in the database directories to locate the database. RDS uses the directory information to try to set up a session with the remote workstation. If the workstation is running, RDS ensures that a session can be established by checking the server, and matching transmission modes and other communications parameters. Now an active session exists between the two LUs. RDS checks conversation security (with user IDs and passwords) to ensure that the user is authorized to have access to the server workstation. If everything is okay, RDS starts a program on the server called the *application agent* to handle the RDS data requests, and to open the database.

If RDS encounters an error at any point in the above scenario, it sends the proper error message back to the application, and the connection process stops. All this is transparent to both the user and the application program. RDS lets the user and application program see no difference between local and remote databases. The mechanism for achieving this is the node and database directories.

NODE AND DATABASE DIRECTORIES

In the OS/2 EE Database Manager, there are three types of directories:

System Database Directory: A master index file for all the databases used on each OS/2 EE workstation. There is only one system database directory on each workstation. It is located in the \SQLLIB\SQLDBDIR\SQLDBDIR file on the same volume as the Database Manager. The file contains one entry for each database stored on the workstation, and one entry for each remote database accessed from the workstation. Each entry in the directory contains a database name and alias, and the type of database--indirect (local) or remote. If the database is local (stored on one of the drives of this workstation), the entry also contains

1. Install and configure both machines, following the information in this article.
2. Start the Communications Manager on both machines.
3. Start the Database Manager on both machines by typing STARTDBM at the command prompt. The Query Manager will do a STARTDBM automatically on the requester when a database is opened.
4. Log on with your user name and password on the requester workstation with a user ID and password that are valid on the server workstation. Passwords permit access to remote workstations. You do not need to log on to the remote workstation. You can log on to the local system by typing LOGON at the command prompt. If you are using Query Manager, this step will be done automatically.
5. Never hardcode the location of the database into your programs. Cataloging is how application programs know the location of the databases. By cataloging the remote workstation and the database, the Database Manager knows where they are. This is usually done as an option in Query Manager, but can also be done via function calls provided by the Database Manager. Cataloging needs to be done only once.

   If a database is moved to a new server, the user must recatalog it. This is much easier than changing the application program.
6. Start using the database. With the Query Manager, the user does this by double-clicking the mouse on the database name or alias.

Figure 1. Steps for Using a Database with RDS

the drive on which it is stored. If the database is remote, the entry contains the name of the remote node on which the database is located. The node name used is one that the user selected; it has no relationship to the workstation name.

Node Directory: Contains the information about remote nodes that will be accessed from the requester workstation. Each workstation contains only one node directory, located in the \SQLLIB\SQLNODIR\SQLNODIR file. There is one entry in the node directory for each LU-LU connection, describing a specific transmission mode. Think of this as one node entry for each workstation with which communication will take place. If the connection is made with APPC and SQLLOO, two node directory entries are needed. Each entry contains the node name (to look up records in the node directory), the local LU used for communications, the PLU profile (containing the address on the network of the remote workstation and the LU on the workstation that is used for communications), and the transmission mode (SQLLOO or APPC).

Volume Database Directory: Exists on each volume with stored databases. If there is one database on C: drive and two on E: drive, there will be two volume database directories on the workstation. The volume database directory is stored in the \SQLDBDIR\SQLDBDIR file on each drive containing databases. One entry is stored in the volume database directory for each database on that drive. Each entry contains the name and alias of the database, the type (always set to HOME), and the OS/2 subdirectory (SQL0001, SQL0002...) in which the database is stored.

With these three directory types, the Database Manager always can locate a database. Consider the following small examples, illustrated in Figures 2 and 3, to see how this works.

Suppose a user is using Query Manager and wants to start using the database ACCOUNTS. The Query Manager calls on the Database Manager and asks it to locate the database. The Database Manager first scans the local system database directory. It finds ACCOUNTS listed as a remote database (on node BANKSRVR), so the Database Manager looks for the remote node BANKSRVR in the node directory. After Database Manager finds it, it passes all relevant information to RDS and then to Com-

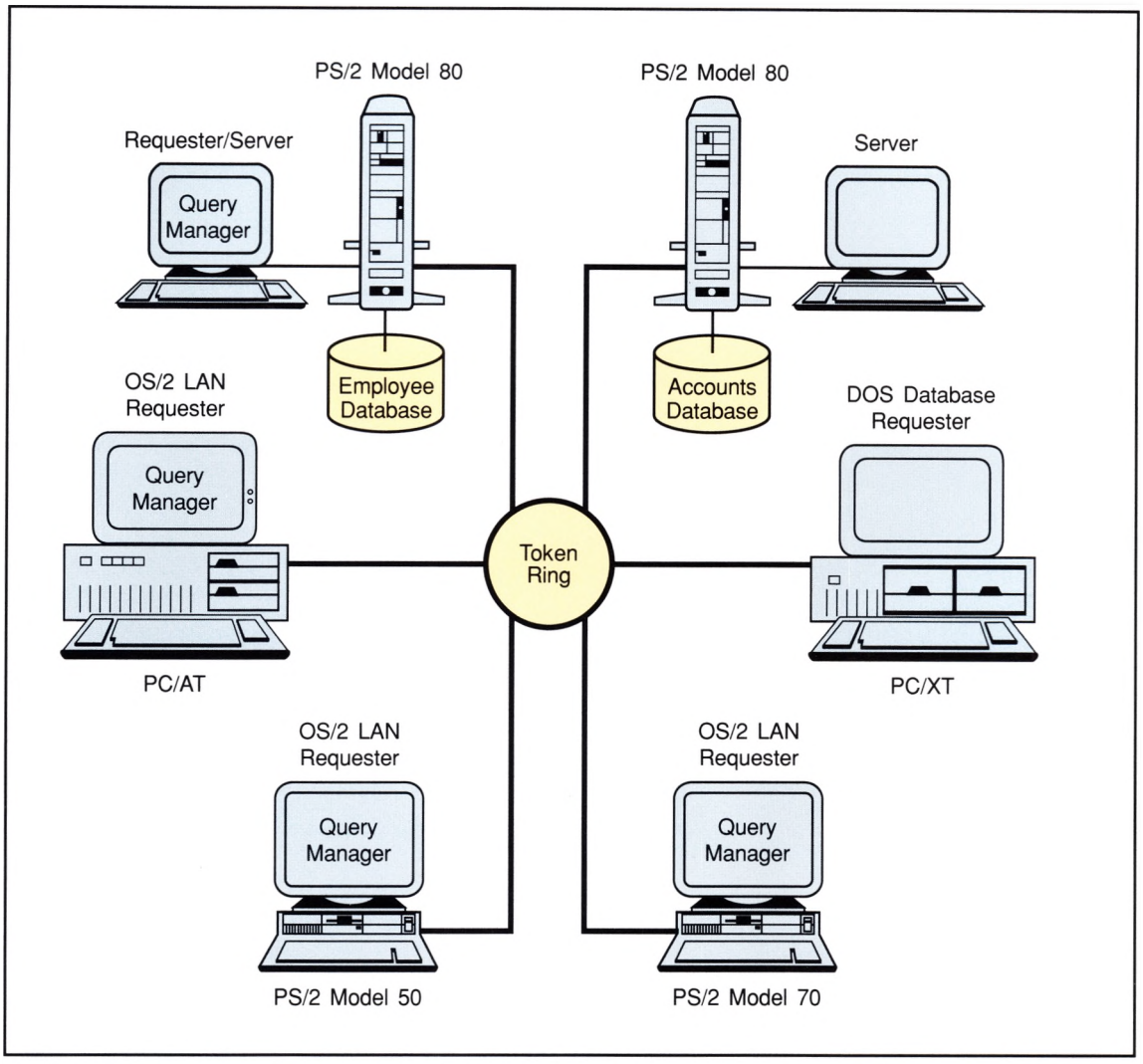


Figure 2. A Sample Bank Network Using RDS

OS/2 EE 1.2 supports one level of remote connectivity.

munications Manager, so that a session can be established to the remote node. Once that has been done, the remote Database Manager looks for the database in its system database directory. It is located on volume D: with a type of INDIRECT. Next, the Database Manager looks in the volume directory for drive D: for the location of the database. Now the Query Manager can open the database and continue.

Assume the user closes database ACCOUNTS and wants to access data in database EMPLOYEE. The Query Manager calls Database Manager. It goes to the system database directory, sees that the database is local (indirect) and then goes to the C: volume database directory. It looks up the OS/2 subdirectory for the database, and opens it.

In both examples, neither the user nor the application knows the location of the database. Usually, database administrators need to know where the databases are located so that they can catalog these databases (for example, enter them into the database and node directories).

OS/2 EE 1.2 supports one level of remote connectivity. If a database is listed in a system database directory as remote, it must be indirect (local) on the remote server. The server's system database directory cannot list the database as remote to that server. This is known as *chaining*. It currently is not supported.

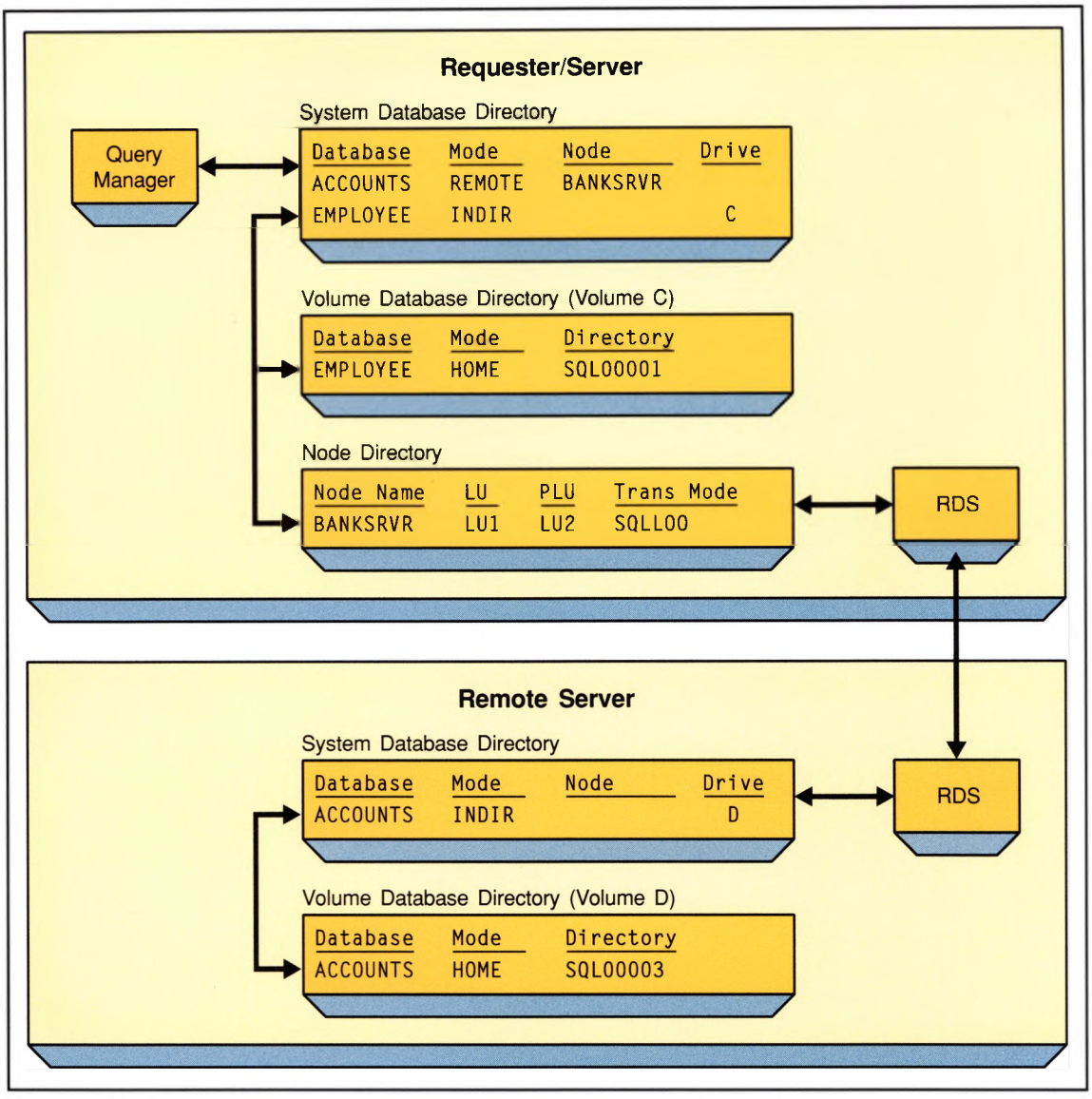


Figure 3. Using the Database Directories

RDS configuration actually consists of installing and configuring the Communications Manager.

INSTALLING RDS

RDS configuration actually consists of installing and configuring the Communications Manager. In fact, any special tailoring of the RDS environment is done almost entirely within Communications Manager. There are several ways to install RDS.

Custom: A custom installation diskette can be made in advance (for example, if many requesters are planned to access the same servers). A custom installation diskette can be installed during the installation of the base OS/2 operating system or using the EEINST program from the OS/2 command prompt.

Advanced Configuration: In the Communications Manager, there is a configuration option under the ADVANCED menu item. This option allows the user to change every feature of the Communications Manager environment, including those parts needed for RDS. Extreme caution should be exercised when using this option. Many fields can be set and changed in the Communications Manager. If an incorrect value is set accidentally in just one of them, the Communications Manager may no longer work properly. Users should exercise caution, since finding and correcting the problem can be very time-consuming.

1. Do you want to create a Basic Configuration File? Yes.
2. The programs ask for the name of the BCS configuration file to create. You can enter an alphanumeric name. The .CFG file will be created in the \CMLIB directory on the drive you use to install the Communications Manager. You can have more than one configuration file, but only one with a given name.
3. The program asks for a comment to describe the BCS file. The Communications Manager displays the comment whenever you can select from a list of more than one configuration file.
4. A panel is displayed with the various BCS features. The one of interest to the database user is #4 RDS.
5. When you select RDS to install, you will be asked if you want your workstation to be a database requester, a database server, or both. If all database servers can also request data from remote databases, then why have both server and server/requester options? If you configure the workstation to be only a database server, BCS will only put server profiles in the configuration file that it is building. This stops your workstation from acting as a requester, although installation as a server causes RDS to install both server and requester programs. You can later add the necessary requester profiles to the configuration file, and then operate as both a requester and server. If you choose to have BCS configure the workstation as a requester/server, it will include both sets of profiles at installation time.
6. Next, RDS asks on what drive you wish to install the Database Manager.
7. The next panel has two questions. The first asks for the type of DLC adapter being used. It is helpful to know what type of DLC you have before installing BCS. If you cannot find the information in the documentation, you can find it on the reference diskette (on PS/2s).

   BCS assumes all Token Ring users have the IBM Token Ring 16/4 cards (16/4/A for PS/2 workstations). If you have a different type of Token Ring card, you may need to use Advanced Configuration to indicate this before using RDS.

   The second question asks if you want to use universal or local LAN addresses. For local addresses, make sure that you get a unique address from your system administrator or make up one you know is unique. If you are using universal addresses, get your designated number before starting. If you cannot get the universal address from your DLC card in advance, you can still install, but you will need to use the Advanced Configuration to put the address in the proper PLU records.
8. If you selected local LAN addresses in the previous panel, RDS asks for the local address of your workstation. This question is not asked for universal addresses because the Communications Manager can get the address directly from the DLC card.
9. If you previously said that your workstation is to be either a requester or a server/requester, you will be asked for the address of the server from which you wish to request data. Remember that BCS only allows a requester to request data from one remote server.

   Note: The BCS file is placed in the \OS2\INSTALL directory, and is moved to the \CMLIB directory after successful installation.

Figure 4. Questions Asked by BCS During RDS Installation

Combination: There is nothing wrong with combining the above choices. In fact, for certain types of operations, such as adding APPC to the Communications Manager configuration profiles or adding multiple server support, it is necessary to use advanced configuration after using BCS or performing a custom install.

BCS

Basic Configuration Services, a program supplied with OS/2 EE 1.2, creates configurations for simplified environments. It also can be run after OS/2 is installed, by running the EECFG program. BCS allows users to answer a few questions and set up their systems. Users who install RDS alone, or RDS and the LAN Requester, might see slightly different numbers in some Communications Manager profiles. These numbers should be of no concern; just be aware that they are different.

BCS builds configuration files only for simple RDS configurations, and can configure remote database access only for SQLLOO (not APPC). BCS can set up only one PLU record for remote requests. A requester that is installed with BCS can initially request data from only one specific server workstation. Use the Advanced Configuration option to use the APPC transmission mode (necessary for an X.25 or SDLC connection), or to set up multiple server support profiles, so that a requester or server can request data from more than one remote server.

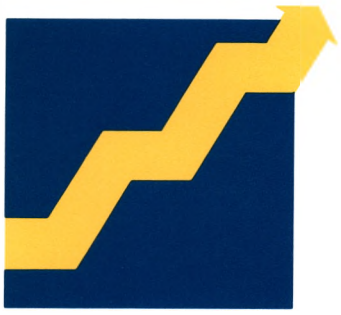
Figure 4 shows questions that must be answered to set up a BCS Communications Manager configuration file for RDS. This is all that is required to run RDS with SQLLOO to one server from each requester. If BCS is not used, the Database Manager section of the installation procedure will ask several of these questions.

When you install the Database Manager, you will be asked if you want to install the DOS database requester. If you are planning to develop DOS database requester applications, you must answer yes to this question on at least one workstation. This copies the files necessary for compiling/linking/running DOS database requester applications to that OS/2 EE workstation.

The Database Manager section of the installation asks for the workstation name. This name is only used if the workstation is a server and will be servicing requests from DOS workstations using the DOS database requester. It must be a unique name, but it has no relationship to the names used to catalog remote nodes.

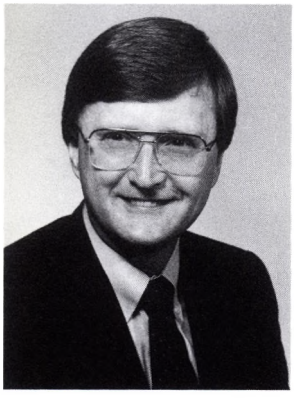
Marty Sirkin, *IBM Entry Systems Division, 11400 Burnet Road, Austin Texas 78758. Mr. Sirkin is a senior associate programmer in the Austin Laboratory and a developer in the Remote Data Services group. He joined IBM in 1989. He received a BS from the California Institute of Technology and an MS in Computer Science from the University of Washington. He is currently on leave from his PhD work at the University of Washington.*

Basic Configuration Services creates configurations for simplified environments.



The Market

Market Update



Bob Tabke

by Bob Tabke

The worldwide market for PC-based multimedia products and services (see Figure 1) is estimated to triple during the next four years, according to Information Workstation Group (IWG) of Alexandria, Virginia. IWG includes video games in their market definition, because video games are expected to evolve over the next five years into very capable multimedia platforms that will be natural add-ons to home information centers. IWG's definition of multimedia does not require motion; presentation packages that incorporate audio are classified as multimedia applications. Figure 2 shows the applications that IWG projects will be important in the U.S. in 1994. IWG plans to update its forecast in January 1991 to reflect emerging applications.

This issue of the *IBM Personal Systems Developer* highlights multimedia tools provided by IBM. According to Allan Ayars, director of the Multimedia Industry Service at Dataquest, "IBM has really sharp people in the multimedia area. They are focused on supporting application development in information access, education, and training. When it all settles, I think we will see that IBM's activity is one of the best-kept secrets in multimedia." Articles in this issue describe the excellent tools that IBM offers. The following sections of this article describe some successful multimedia applications today, and how to get the most from IBM's support programs.

The worldwide multimedia market is estimated to triple during the next four years.

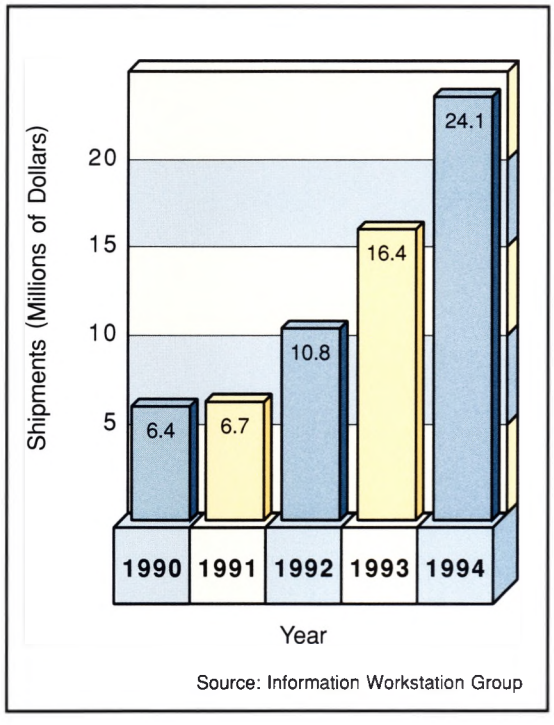


Figure 1. Worldwide Multimedia Market

EDUCATION

Figure 3 lists ten benefits of multimedia technologies for education. The advantage of multimedia courseware is that it is highly interactive and engages the learner. This motivates learning, maintains a high level of attention, and results in increased retention. For example, courseware now available from Magnus Communications Design teaches high school students across the country about sexually transmitted diseases. Because standards vary across the U.S. regarding acceptable topics for discussion in school, the application provides a parameter that determines how explicit or vague the course material will be by controlling which video clips are shown. In one segment of this course, teenagers are at a party at a student's house when the student's parents are out of town. At different points during the party, the boy or girl in the video clips must make decisions. The learner makes these decisions and the application plays the next appropriate video clip. There are several scenarios for each

decision path, so that if the course is played more than once it does not repeat itself. This gives the impression of spontaneity. Educators who have previewed this course are fully satisfied with it. It deals with a sensitive, complex subject, yet it keeps the teenagers interested. Because the course requires the learner to participate, it is almost like experiencing that situation—with the ability to explore the various possible outcomes. For more information, contact Magnus in Vancouver, B.C. at (604) 682-2266.

TARGET, a predecessor to the sexually transmitted diseases program, deals with alcohol and drug abuse. It has been very successful. The program is described in an IBM Application Brief (GK30-0055).

Multimedia encyclopedias are excellent examples of how multimedia can convey information more effectively and efficiently than text and graphics alone. Multimedia encyclopedias use sound to convey additional information. If you look up a mockingbird, the definition includes a sound track of a mockingbird. If you look up Martin Luther King, you can hear a portion of one of his famous speeches. Multimedia encyclopedias provide a type of news report. A tremendous amount of information can be conveyed in this way. Because the video and sound are compressed and the clips are short, they can be delivered economically on CD-ROM.

Learning a foreign language often makes students feel confused and anxious, particularly if it differs greatly from their native language. In a successful effort to promote better learning and understanding of Hebrew, the University of Michigan uses a multimedia application that presents a courtroom drama in Hebrew and English, with a variety of built-in learning tools. The educators chose the courtroom scenario for several reasons. It is the simulation of a real situation in which the student plays the role of the judge. This requires student action and involvement. Each of four witnesses is examined and cross-examined (there are eight versions of the facts in the case). The story approach helps to relax students. The student who plays the role of judge must understand all the testimony and make a decision at the end of the program. Several on-screen dictionaries aid in learning. One is visual, giving language explanations in either Hebrew or English. Another dictionary allows the student to touch any word on the screen for a translation. Because the courtroom drama uses native Hebrew actors, students learn more than just the language—they hear different accents and word usage.

Distributing consistent, quality information to diabetics is easier with a new multimedia application developed at Auburn University. A bedside unit lets patients access information about warning signs, adult testing, daily medication, diet planning, and proper exercises. Participants repeat selections as often as they like. The system also documents and evaluates the patient's understanding, which helps the doctor.

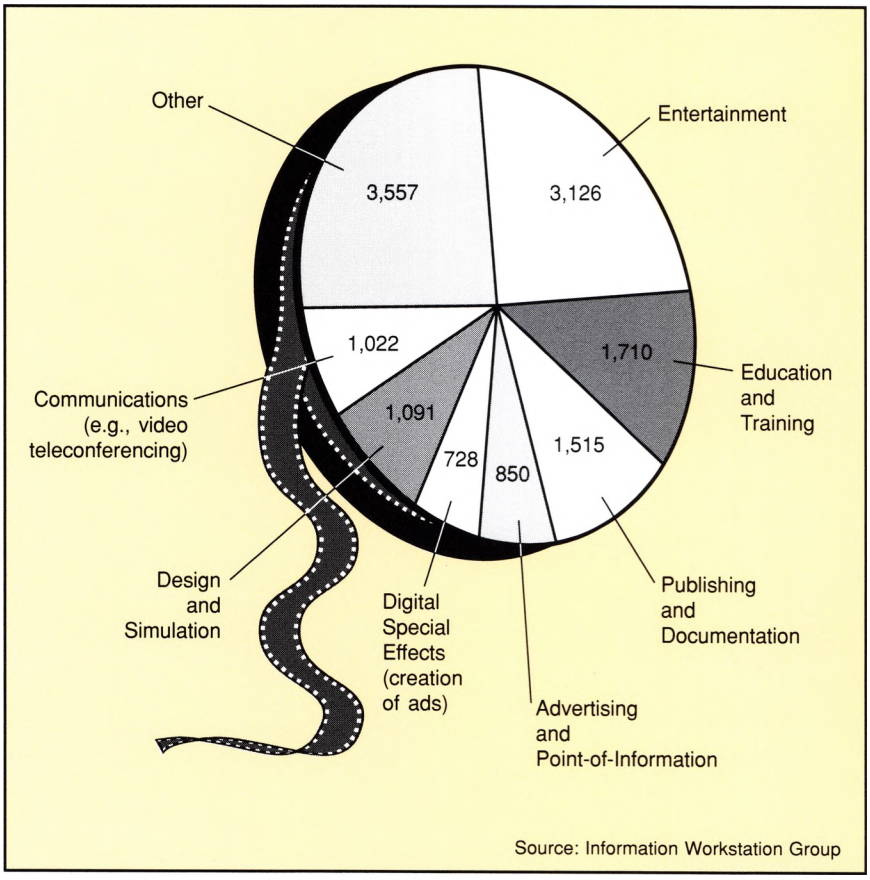


Figure 2. Major Multimedia Applications in 1994 (U.S., millions of dollars)

Chemistry students at the University of Illinois are conducting lab experiments using interactive multimedia technology. Multimedia makes complex chemical reactions easier to understand since abstract concepts can be illustrated. Students mix chemicals on the screen and then observe the reactions in detail using slow motion or advancing the action step-by-step. Students can experiment with more reactions in less time. This reduces

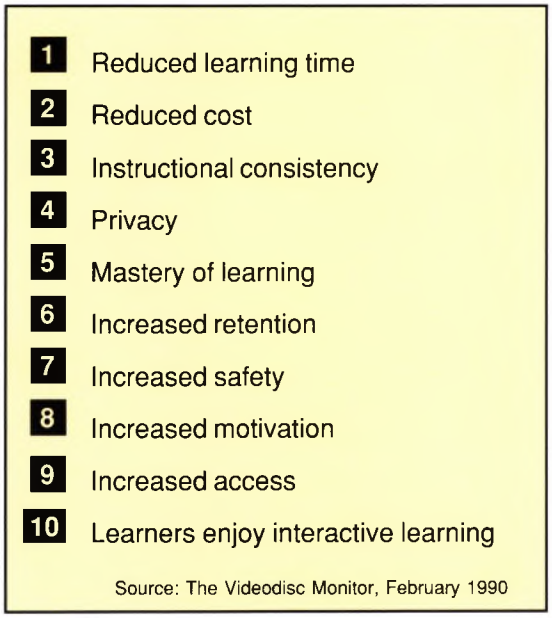


Figure 3. Benefits of Multimedia in Education

The IBM Multimedia Courseware Catalog lists nearly 600 products.

costs by eliminating chemicals and equipment. It focuses the student's attention on the correct observations, thereby enhancing effectiveness. The computer is much safer to use than the real chemicals. Details of this application are available in an IBM Application Brief (GK21-0074).

The concept behind "just-in-time" training is to provide automated reference material to workers in a convenient form that they can use before engaging in a complex task. It is especially advantageous where workers must remember details on many different complex topics. For example, since it is difficult for commercial airline mechanics to remember all the information about the various aircraft they maintain, they must rely on reference books. Multimedia planners visualize kiosks located in the aircraft maintenance area for the mechanic's reference before working on an unfamiliar part. The mechanic can get information about that particular revision level of a part, perhaps including a video clip showing normal operation and how to perform the repair. Since the kiosks can be sealed, they can be operated in a hostile environment.

Chase Manhattan Bank is using multimedia touch-screen technology to train tellers to perform routine transactions such as check handling and customer service activities. New employees work through several realistic workplace simulations. It heightens a trainee's interest and motivation to see different transactions progress from beginning to end.

IBM is one of the world's largest users of multimedia computing. Each year, IBM trains thousands of employees and customers at over 160 learning centers across the U.S. Marketing education is a major emphasis of this training. Interactive programs introduce students to the fundamentals of different industries. These programs expose them to typical situations they might encounter in sales assignments. This approach is extremely cost effective. Learners who are actively involved can absorb more information in less time.

In 1987, IBM helped seventeen medical and nursing schools to form a self-governing organization—the Healthcare Interactive Videodisc Consortium (HCIVC)—to develop interactive videodisc courseware. Several applications have been completed and described in a booklet by IBM (GK21-0103).

The IBM Multimedia Courseware Catalog lists nearly 600 products ranging from $1,000 to over $4,000 (ordering information is mentioned later in this article). At these prices, the products are not designed for mass distribution. They can be used effectively in field locations where several people can take advantage of the material. One great advantage of multimedia courses is that they can be delivered to the job site; workers do not have to leave the field to take a course. This significantly reduces the productivity impact of training and also cuts travel costs.

PRESENTATION TOOLS

Although decisions are often based on a presentation, decision makers often do not fully understand everything that was presented to them. The goals of multimedia presentations are information understanding and retention. The group receiving the presentation gets more information in less time with multimedia. Most important is that the decision makers (and those that must support the decision) feel that they really understand the matter under consideration. Multimedia tools permit business people to create presentations that use animation, sound, and graphics. As Dataquest's Allan Ayars observed, "Multimedia presentations do not have to be as fancy as the 6:00 PM news. All that is needed is the ability to lay down a sound

track, a track for video and some special effects, and a voice track." This can be done with a PC or delivered by video tape. The range of potential users is broad.

Making an effective presentation involves taking a concept that has been developed over many months and translating it into words or transparencies to communicate it to others. Receivers must understand the words and recreate (within their own context) the presenter's concept. By using as many attributes as possible (such as pictures, animation, and sound), the recipient can more accurately reconstruct the concept. An old axiom states that people retain only 20% of what is presented to them. As the presenter, you would like to control the material that is included in that 20%. One way to do that is to highlight the most important information by using animation, video, or sound.

For example, a sales manager may simply want to show how a sales territory will evolve over time: this year he is going to open Atlanta, next year Birmingham. A multimedia presentation allows the audience to watch the dramatic blending of those two territories. They can see one portion of the southeastern region become a different color as Birmingham opens and spreads all the way to Mobile. Even the most casual listener will remember more from this kind of presentation than one based on transparencies or 35 mm slides.

A software application called Podium, developed at the University of Delaware, provides a quick and easy way to show slides, display animation sequences, play audio, and project video. To prepare for a presentation, the lecturer uses any word processor to create an outline of the multimedia objects to be displayed. Podium reads the outline and prepares the presentation. Lecturers can quickly prepare a custom presentation from previously saved images—even the night before the presentation. Instructors are finding that Podium allows them to interact effectively with their classrooms by enabling them to move forward or backward to show media sequences that relate to students' questions. Further details are available in an IBM Application Brief (GK21-0100).

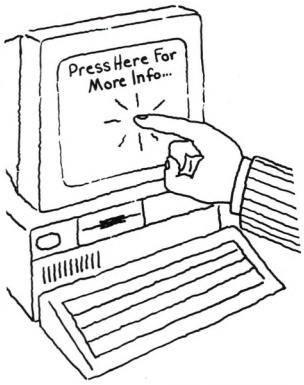
INFORMATION ACCESS

Touch-screen technology can provide a simple user interface, suitable for use by anyone. A basic building block for interactive public information access systems, touch can be inexpensively added to a standard CRT or a special CRT can be used. These systems can relieve inquiry centers in government and business from repeatedly answering common questions.

One of the most used information access systems was at Expo '86 in Vancouver, where over ten million visitors used IBM's EXPO INFO touch-screen system. "I've finally found a computer system I can use!" was the comment most often recorded from those visitors who were asked for an opinion. Since then, BC Systems has produced several public access applications for information belonging to the government of British Columbia. For more information, contact Donald Spears, director of the InfoBC Project, at (604) 389-3101.

A county government has put information kiosks in shopping malls to inform the public of available services ranging from auto licensing to upcoming events. With fingertip access, information is easy to obtain and use. Hard-copy printouts are available so that the user can take information and use it later.

Touch-screen technology can provide a simple user interface, suitable for use by anyone.



MERCHANDISING

Herman's Sporting Goods uses a multimedia merchandising application to inform customers about products and resort destinations where activities like golf, tennis, skiing, and water sports are popular. Users can even receive a printout of directions and accommodations. The program provides sporting-goods manufacturers and resorts with an economical way to merchandise their products. It also provides shoppers a convenient way to explore vacation destinations and obtain information about products.

Long and Foster, a real estate firm in the Washington D.C. area, is using multimedia to selectively present homes to prospective buyers. The system provides detailed information on the neighborhood, including schools and other features important to a prospective home buyer. The system even calculates mortgage payments and closing costs.

The best place to get information is IBM's Multimedia Customer Assistance Center.

OTHER APPLICATIONS

Nevada's Department of Motor Vehicles uses a multimedia application to qualify applicants for a commercial driver's license. The motion video, graphics, and audio help applicants to better comprehend information. Video scenarios place applicants in actual driving situations. Applicants respond with their decisions and the computer scores their responses. This saves time for both the applicants and clerks.

Sears Catalog Division uses multimedia for their 400 merchandise buyers to check catalog items in addition to their sales, stocking, vendor, and ordering data. The online system replaces the manual process of tearing out catalog pages and creating sales spreadsheets. The buyer can simultaneously see a photo of the item and all information relating to it.

IBM SUPPORT AVAILABLE

The IBM Multimedia Solutions group began as IBM's internal education development group. As customers saw the videodisc-based courseware developed by the group, they began to ask for tools so that they could implement similar programs. During the past four years, the group has developed commercial tools and multimedia hardware, cultivated third-party developers, and created a market. The following sections describe the multimedia support available from IBM.

Telephone Support

There is currently no formal IBM program for multimedia developers. The best place to inquire about support, ask technical questions, and get product information is IBM's Multimedia Customer Assistance Center in Atlanta, (800) 627-0920.

Bulletin Board System

IBM provides an electronic Bulletin Board System (BBS) that allows developers or users of LinkWay and the Audio Visual Connection (AVC) to communicate with each other about uses, developing applications, and exchanging tips and techniques. The system allows script buttons and public domain applications to be uploaded and downloaded. IBM also posts announcements on the BBS. The BBS is free and can be reached at (800) 247-3461 (set communications software to N-8-1).

Courses

IBM offers several classes (in Atlanta, Georgia) about its products for customers. Information and class schedules can be obtained by calling the Multimedia Customer Assistance Center.

Newsletter

The *IBM Multimedia Solutions* newsletter (monthly) contains product information, applications descriptions, and helpful tips. Anyone can subscribe by contacting Ms. Rhonda Rattray, IBM Multimedia Solutions, P.O. Box 2150, Atlanta, GA 30301.

Courseware Catalog

IBM provides a *free* catalog of nearly 600 interactive educational and instructional courseware products. It can be obtained by calling the IBM Customer Assistance Center or Learning Ventures International (LVI) at (800) 344-3115. LVI also publishes *Human Capital*, a monthly magazine for executives, available for $29.95 annually. It claims to take the technology out of multimedia and present its benefits so that senior managers can evaluate its potential for improving their organization's operations.

Society is going to be profoundly affected by multimedia technology. It will be used by children in school, workers on the assembly line, business presenters and information workers, accountants, and government employees. Nearly every aspect of society is a potential beneficiary of this technology.

Bob Tabke, *Senior Associate, Dataquest, Inc., 1290 Ridder Park Drive, San Jose, CA 95131. (415) 493-4352. Mr. Tabke is a consultant to Dataquest in the personal computer and telecommunications areas. He has a BS in Electrical Engineering from Purdue University and an MBA from Harvard University.*

© IBM Corporation
All Rights Reserved

International Business Machines Corporation
P.O. Box 1328
Boca Raton, FL 33429-1328

G362-0001-07